FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Sistema avançado de configuração e simulação de unidades fabris

João Pedro de Jesus Barbosa Pinto

Mestrado Integrado em Engenharia Informática e Computação

Orientador: Rosaldo José Fernandes Rossetti (Prof. Doutor)

Julho de 2014

Sistema avançado de configuração e simulação de unidades fabris

João Pedro de Jesus Barbosa Pinto

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: Carlos Manuel Milheiro de Oliveira Pinto Soares (Prof. Associado)

Vogal Externo: Artur José Carneiro Pereira (Prof. Auxiliar)

Orientador: Rosaldo José Fernandes Rossetti (Prof. Auxiliar)

25 de Julho de 2014

Resumo

A simulação é uma das mais poderosas ferramentas disponíveis hoje em dia para indústrias que desejam antecipar possíveis situações ou mudanças de teste no funcionamento dos seus processos de trabalho. Um Sistema para a Execução de Manufatura (MES) não é nada mais do que uma ferramenta que permite um melhor controlo sobre esses mesmos procedimentos, fazendo uma ponte entre os sistemas de informação abstratos de alto nível e os sistemas que controlam as máquinas na linha de produção.

O objetivo inicial deste projeto foi pesquisar simuladores atualmente no mercado que fossem capazes de simular uma ferramenta chamada cmNavigo, um MES. Uma vez que nenhum simulador com esta capacidade foi encontrado, o objetivo principal passou pela pesquisa e entendimento de como seria possível implementar um simulador para esta ferramenta.

Com este projeto foi possível perceber como é que se poderia conceber um modelo de simulação, e posterior implementação desse mesmo modelo, de um Sistema para a Execução da Manufatura. Foi realizado um estudo sobre os principais conceitos da simulação e dos trabalhos relacionados na área para delinear a melhor estratégia a seguir e foi possível concluir que apesar dos principais simuladores industriais atualmente existentes no mercado seguirem uma abordagem de Modelação por Eventos Discretos (DEM), a tendência emergente é a da Simulação Baseada em Agentes (ABS) já que esta, teoricamente, consegue obter melhores resultados a nível global. Tendo isso em conta uma abordagem para resolver o problema foi desenvolvida e implementada, misturando a DEM padronizada com a nova ABS.

A solução foi dividida em duas grandes etapas: o modelo é criado em primeiro lugar, através de uma aplicação, e a simulação é realizada de seguida por outra. Uma aplicação final também foi desenvolvida para permitir que o utilizador seja capaz de extrair os resultados do cmNavigo quando a simulação termina. O modelo é criado através da personalização dos dados para a simulação pelo usuário na primeira aplicação. Quando está satisfeito, pode então simular o comportamento de cmNavigo numa fábrica virtual.

No final, foi possível concluir que uma vez que este é um ramo pouco explorado da simulação, os únicos resultados que se podem obter são sobre a utilidade do modelo DEM + ABS e sobre o sucesso de uma simulação correta, comparando execuções manuais de cmNavigo com resultados do próprio simulador.

Abstract

Simulation is one of the most powerful tools available today for industries wishing to anticipate possible situations or test changes in the functioning of their work processes. A Manufacturing Execution System (MES) is nothing more than a tool that allows a better control over these same procedures, making a bridge between the abstract high-level information systems and the systems that control the machines on the production line.

The initial objective of this project was to research simulators currently on the market that were able to simulate a tool called cmNavigo, which is a MES. Since no simulator with this capability was found, the main goal went through search and understand how it would be possible to implement a simulator for this tool.

With this project it was possible to understand how one might design a simulation model, and subsequent simulation of that model, of a MES. A study of the major concepts of simulation and related work in the area was carried out to delineate the best strategy to follow and it was concluded that despite the major industrial simulators currently on the market follow a Discrete Event Modeling (DEM) approach, the emerging trend is the Agent Based Simulation (ABS) since this can, theoretically, obtain best results globally. Taking this into account, an approach to solve the problem was developed and implemented, mixing the standardized DEM with the new ABS.

The solution was divided into two big steps: the model is created first, through one application, and the simulation is performed after by another one. A final application was also developed to allow the user to be able to extract the results from cmNavigo when the simulation ends. The model is created through the customization of the data for the simulation by the user in the first application. When he is satisfied, he can then simulate the behavior of cmNavigo in a virtual factory.

In the end, it was possible to conclude that since this is an unexplored branch of the simulation, the only results we might have are towards the utility of the DEM + ABS model and the success of a correct simulation, comparing manual executions of cmNavigo with results of the simulator itself.

Agradecimentos

Ao meu orientador, o Professor Rosaldo Rossetti, por me ajudar a entender melhor um tema que me era desconhecido e por me dar a conhecer métodos de trabalho sem os quais não conseguia ter feito este projeto.

Ao João Brandão, o meu orientador na Critical, que se mostrou sempre disponível para trocar ideias comigo.

Aos meus amigos, sem eles estes últimos 5 anos não tinham sido o mesmo.

E especialmente há minha família, sem eles não seria quem sou e não estaria a escrever estas palavras.

O meu obrigado,

João Pinto

“Imagine if every Thursday your shoes exploded if you tied them the usual way. This happens to us all the time with computers, and nobody thinks of complaining.”

Jef Raskin

Conteúdo

Introdução **1**
1.1 Contexto/Enquadramento 1
1.2 Formalização do Problema 2
1.3 Motivação 3
1.4 Objetivos 4
1.5 Metodologia 5
1.6 Contribuições Esperadas 6
1.7 Estrutura do Relatório 6

Revisão Bibliográfica: Background **9**
2.1 Simulação 9
2.1.1 Definição/Conceitos 9
2.1.2 Modelos 11
2.1.3 Paradigmas 13
2.2 Manufacturing Execution System 21
2.2.1 Definição e Funcionalidades 22
2.2.2 Tecnologias 23
2.2.3 Arquitetura 24
2.2.4 Atuais Limitações 24
2.2.5 Futuro dos MES 25
2.3 cmNavigo 26
2.3.1 Arquitetura 27
2.3.2 Módulos 28
2.4 Sumário 30

Revisão Bibliográfica: Trabalho Relacionado **31**
3.1 Modelação de MES 31
3.1.1 Agent-based modeling and simulation of an autonomic MES 31
3.1.2 Integration of Manufacturing Execution System and Simulation 34
3.1.3 Modeling and Simulation Approach for an Industrial MES 36
3.2 Simulação de MES 38

3.2.1 From Conceptual Models to Executable Simulation 39
3.2.2 Simuladores Industriais 41
3.3 Sumário 42

Abordagem Metodológica 43
4.1 Introdução 43
4.2 Escolha da Solução 44
4.2.1 Abordagem Escolhida 44
4.2.2 Tecnologias 46
4.2.3 Trabalho Realizado 46
4.3 Verificação e Validação 47
4.4 Sumário 49

Implementação 52
5.1 cmNavigo 52
5.2 Master Loader 54
5.2.1 Master Data 55
5.2.2 Aplicação 56
5.3 *cmSimulatorConfig* 57
5.3.1 Aplicação 58
5.4 *cmSimulator* 62
5.4.1 Lógica 62
5.4.2 Aplicação 66
5.5 Sumário 69

Resultados e Discussão 70
6.1 Validação e Verificação 72

Conclusão e Trabalho Futuro 74

Referências 76

Anexo A: Planeamento 81

Anexo B: Principais Simuladores 83

Anexo C: Review Planning 90

Anexo D: Relatório de Requisitos 93

Anexo E: Relatório de Arquitetura 100

Anexo F: Relatório de Tecnologias 111

Anexo G: Objetos do cmNavigo 120

Lista de Figuras

Figura 1.1 e 1.2: A atual função do cmNavigo e o que poderá fazer com o simulador 3
Figura 2: Esquema de um Estudo de Simulação 10
Figura 3: Principais Paradgimas da Simulação existentes 13
Figura 4: Diagrama de Laço Causal 14
Figura 5: Funcionalidades de um Manufacturing Execution System 22
Figura 6: Arquitetura básica de um MES 24
Figura 7: Arquitetura Lógica do cmNavigo 27
Figura 8: Arquitetura Física do cmNavigo 28
Figura 9: Metodologia Prometeus combinada com a abordagem Hermes 32
Figura 10: Modelo utilizando dois tipos de agentes 33
Figura 11: (A) Separação da modelação e (B) Linha de produção de aço 35
Figura 12: Modelo da Linha de Produção de Aço 36
Figura 13: Resultados da simulação efetuada 37
Figura 14: Diagrama de Estados do MES 38
Figura 15: Da concepção à simulação 40
Figura 16: Ambiente de desenvolvimento do Simul8 41
Figura 17: Screenshot da ferramenta fabLIVE 46
Figura 18: Validação de uma aplicação de simulação 48
Figura 19: Modelo Relacional dos principais objetos do sistema 53
Figura 20: Folha do MasterData correspondente aos *Resources*. 55
Figura 21: Uma utilização do MasterLoader 56
Figura 22: Uma utilização do Master Loader 57
Figura 23: Aba de escolha dos Produtos a serem usados na simulação 58
Figura 24: Configuração dos Materials usados na simulação 59
Figura 25: Escolha dos Resources a serem testados 60
Figura 26: Steps possíveis de um Material efetuar Rework 61
Figura 27: Interface do cmSimulator 67
Figura 28: Visualização da Falha de Resources no fabLive. 67
Figura 29: Histórico detalhado de um Material. 68
Figura 30: Planeamento do Trabalho 82

Lista de Tabelas

Tabela 1: Características dos principais Paradigmas 20
Tabela 2: Resultados da Simulação 71
Tabela 3: Principais simuladores de MES 1/6 84
Tabela 4: Principais simuladores de MES 2/6 85
Tabela 5: Principais simuladores de MES 3/6 86
Tabela 6: Principais simuladores de MES 4/6 87
Tabela 7: Principais simuladores de MES 5/6 88
Tabela 8: Principais simuladores de MES 6/6 89
Tabela 9: Objetos que pertencem ao Master Loader 120

Abreviaturas e Símbolos

ABS	Agent Based Simulation
CM	Critical Manufacturing
CRUD	Create, Read, Update, Delete
CS	Critical Software
DLC	Diagrama de Laço Causal
ENIAC	Electronic Numerical Integrator And Computer
ERP	Enterprise Resource Planning
FEUP	Faculdade de Engenharia da Universidade do Porto
FML	Flexible Manufacturing System
GABS	Generic Agent-Based Simulation
HMS	Holonic Manufacturing System
IGs	Interaction Goals
MABLe	Modeling Agent Behavior by Learning
MES	Manufacturing Execution System
MESA	Manufacturing Enterprise Solutions Association
OEE	Overall Equipment Effectiveness
OLAP	On-Line Analytical Processing
PRE	Planeamento dos Recursos da Empresa
RMES	Reconfigurable Manufacturing Execution System
SD	Sistemas Dinâmicos
SEM	Sistemas para a Execução da Manufatura
SPC	Statistics Process Control
SWOT	Strengths Weaknesses Opportunities Threats
UPH	Unit Per Hour

Capítulo 1

Introdução

Este é o capítulo introdutório do relatório. Aqui será feito o enquadramento do projeto, uma formalização do problema para conseguir deliniar quais são os objetivos a cumprir, objetivos estes que também estarão especificados neste capítulo. É ainda explicada a motivação científica e pessoal para a realização do projeto, a metodologia adotada para o resolver e as contribuições que são esperadas para o futuro desta área com a realização do mesmo. No fim é apresentada a estrutura do documento.

1.1 Contexto/Enquadramento

A Critical Manufacturing é uma *spin-off* da Critical Software que foi fundada em 2009 e está focada no desenvolvimento de *software* de automação e produção para indústrias de alta tecnologia, tais como a da energia fotovoltaica, eletrónicos e semicondutores.

Um dos principais *softwares* desenvolvidos pela Critical Manufacturing tem por nome cmNavigo. Este produto é um Sistemas para a Execução da Manufatura (*Manufacturing Execution System*) totalmente baseada em plataformas e tecnologias Microsoft, extremamente avançado e que permite ao utilizador saber se o que foi planeado pelo sistema ERP (*Enterprise Resource Planning* ou Planeamento dos Recursos da Empresa) está a ser corretamente executado, e se não o está o porquê disso acontecer, onde estão a ocorrer os problemas e até o tipo de problemas que ocorrem mais frequentemente.

Estando este *software* em concorrência direta com alguns do produtos de topo a nível mundial, e dada a juventude da solução, necessariamente com uma base de clientes de referência inferior, é necessário que esta apresente características inovadoras e diferenciadoras. Este projeto consiste no desenho e criação de uma solução de configuração e simulação do

cmNavigo para testes, demonstração a clientes e provas de conceito, sendo que o resultado final se traduzirá numa solução que trará grandes vantagens competitivas à empresa.

Nos dias de hoje, a simulação é uma das ferramentas indispensáveis e mais poderosas que está disponível para os responsáveis pelas decisões mais importantes de uma empresa, já que os auxilia no desenho e no funcionamento dos processos e sistemas mais complexos e torna possível analisar e avaliar situações que não seriam possíveis de testar sem se comprometerem primeiro com as mudanças em estudo. Num mundo tão competitivo como é o mundo atual, a simulação tornou-se indispensável na resolução de problemas das mais variadas áreas. [SRE98]

1.2 Formalização do Problema

O problema que se tenta resolver prende-se com a necessidade deste *software* já existente precisar de uma componente: um simulador. Pode-se definir então como a

Falta de um simulador apropriado que se integre totalmente com uma solução já desenvolvida na empresa (cmNavigo) que seja rápido de configurar, leve, fácil de usar e possa trazer vantagens competitivas ao cliente.

Neste caso, a divisão do problema em duas fases foi a melhor solução. As duas fases podem-se então definir pela fase de investigação e pela fase de desenvolvimento. Na primeira fase houve uma pesquisa sobre os simuladores já existentes e que são usados em indústrias semelhantes pelos respetivos Sistemas para a Execução da Manufatura. Assim levantam-se três questões de investigação extremamente relevantes:

Quais são os simuladores de Sistemas para a Execução da Manufatura atualmente disponíveis e usados principalmente pelas indústrias semelhantes?

Conseguindo obter resposta a esta pergunta, tornar-se-á mais claro que tipo de requisitos terá que ter o simulador a ser desenvolvido. Muitos deles serão semelhantes já que o que é simulado seria um *software* de Sistemas para a Execução da Manufatura diferente do cmNavigo.

Se realmente existem simuladores, são capazes de cumprir todos os requisitos da Critical Manufacturing para este tipo de aplicação?

A Critical procura ter alguns requisitos muitos específicos incluídos na solução final, de maneira a que esta se consiga distinguir do resto dos produtos presentes no mercado:

- Permitir uma rápida modelização de linhas de produção piloto;

- Permitir a cópia de modelos entre sistemas diferentes;
- Efetuar a simulação da operação do sistema;
- Visualizar em tempo real o sistema em funcionamento;
- Armazenar dados da simulação para efeitos de provas de conceito, relatórios e análises;
- Efetuar testes de *stress* ao sistema.

Se algum dos simuladores conseguir cobrir todos os requisitos, a estratégia passaria apenas por adaptar o cmNavigo ao simulador já existente, diminuindo bastante o trabalho prático a ser realizado.

Se nenhum dos simuladores conseguir cumprir todos os requisitos, como conceber e implementar um que ao mesmo tempo consegue preencher todos os requisitos propostos e que seja leve e mais fácil de usar do que os atualmente conhecidos?

Caso nenhum dos simuladores atualmente no mercado consiga satisfazer todos os requisitos requeridos pela CM, será necessário desenvolver um de raiz. Assim a pesquisa necessária terá também que englobar o desenvolvimento de *softwares* de simulação. Além de todos estes requisitos, uma das maiores diferenciações neste projeto é fazer com que o cliente tenha o mínimo de trabalho possível ao inserir os dados da sua linha de produção no simulador. Esta será uma das grandes vantagens competitivas em relação aos *softwares* atualmente líderes de mercado.

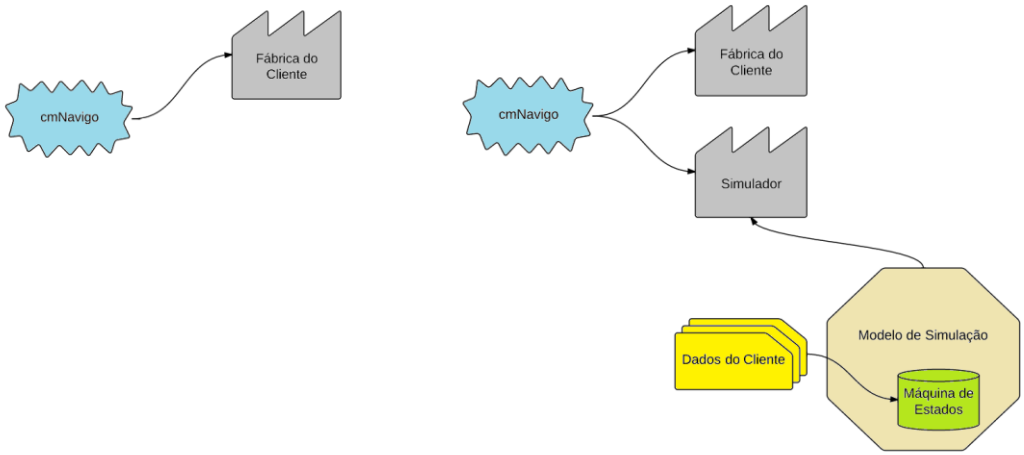


Figura 1.1 e 1.2: A atual função do cmNavigo e o que poderá fazer com o simulador

1.3 Motivação

O mundo atual é um mundo onde a mudança é constante, onde os mercados globais estão em permanente evolução e onde todos querem ter lucro de uma maneira ou de outra. Assim, as

empresas e organizações têm de se mostrar mais inovadores e flexíveis, tendo ao mesmo tempo que manter qualidade e tempos de resposta reduzidos para se manterem competitivas. Estas normas não são verdadeiras apenas para o produto final mas também para os processos de produção. Aliado à grande dimensão e à elevada complexidade dos sistemas de produção atuais, torna-se imprescindível o desenvolvimento de ferramentas que permitam avaliar e otimizar os processos produtivos existentes.

A simulação é uma das melhores formas de analisar o desempenho das situações atuais de cada empresa e avaliar as possíveis soluções que se tenham em mente, com o objetivo final de melhorar os recursos e os processos atualmente estabelecidos na organização. É então bastante relevante que qualquer gestor de uma empresa tenha acesso a uma ferramenta personalizada que o auxilie nas principais tomadas de decisão que impliquem alterações significativas nos processos da sua organização, decisões estas que têm custos elevados e grandes impactos na forma como é fabricado o produto final.

Com o crescimento da complexidade dos sistemas e dos processos industriais, é ainda mais imperativo recorrer primeiro a uma simulação antes de se comprometer com a aplicação prática das soluções idealizadas. Através desta simulação, é possível estabelecer as melhorias que se podem obter, bem como detetar alguns dos erros que iriam ocorrer, antes da implementação concreta de um novo sistema ou de uma pequena mudança que se queira testar.

A realização deste trabalho irá permitir aprofundar conhecimentos numa área pouco difundida na FEUP - a modelação e simulação de sistemas, apesar de nos dias que correm a simulação já ser indispensável no mundo industrial. Será de grande contribuição científica melhorar o processo de recolha de dados de um cliente pois irá facilitar a futuros projetos a criação de um simulador que consuma menos recursos, assim como criar um simulador que vá de encontro aos requisitos de um *software* já existente no mercado.

1.4 Objetivos

O objetivo geral desta dissertação passa pelo estudo e conceção de um simulador de um Sistemas para a Execução da Manufatura, delineando a melhor estratégia para o desenvolvimento do modelo da simulação. Além disto, este simulador será ainda implementado de forma a conseguir integrar o *software* da CM - cmNavigo.

Dentro deste objetivo geral, são possíveis destacar ainda alguns objetivos mais específicos:

- Estudo e elaboração de um relatório sobre os conceitos e definições que irão ser necessárias para a elaboração do projeto, assim como do estado da arte ao nível dos trabalhos relacionados;
- Estudo da solução desenvolvida pela CM - cmNavigo;

- Levantamento dos requisitos e respetivo desenho da arquitetura para a solução final;
- Agilização do processo de levantamento dos dados do cliente para a criação do modelo de estados a usar na simulação;
- Conceção de um modelo de simulação que integre da melhor maneira os diferentes paradigmas;
- Desenvolvimento de um protótipo de um simulador;
- Teste do protótipo com dados de teste e dados reais;
- Análise dos resultados obtidos;
- Escrita da dissertação.

1.5 Metodologia

O trabalho de investigação inicial será crucial no rumo a seguir. Dependendo dos resultados da investigação efetuada, este projeto poderá seguir dois caminhos bastante distintos: já existe um simulador que cumpra todos os requisitos propostos pela CM ou não existe um *software* tão complexo que consiga cobrir os requisitos mais específicos. Assim, a primeira parte deste projeto focar-se-á numa pesquisa de todos os produtos existentes no mercado que eventualmente poderiam ser usados pelo cmNavigo para simular uma linha de produção industrial, de acordo com uma *review planning* apresentada no Anexo C. Concluída esta pesquisa, poder-se-á optar por um de dois caminhos diferentes.

No primeiro caso, onde existe efetivamente um simulador no mercado, ou até um protótipo de um simulador que eventualmente se poderia completar, o rumo do projeto será agilizar o processo de obtenção dos dados do cliente, de maneira a que este consiga simular a sua linha e produção da forma mais eficiente possível, recorrendo apenas a um punhado de dados para cada estado. É de importância referir que estes dados serão sempre *inputs* do cmNavigo, por isso a escolha do simulador nunca iria afetar esta parte do projeto. Para isso, a investigação focar-se-á na estrutura das diferentes linhas de produção das indústrias de alta tecnologia de maneira a encontrar um padrão simples que se consiga adaptar ao máximo número de componentes dessa mesma linha de produção. Por exemplo, detetar o padrão dos dados mestre necessários para conseguir criar um estado que simule qualquer máquina. Um exemplo para estes dados seria:

- X + Y + Z = W, onde X, Y e Z são as unidades de vários tipos de matéria-prima que são recebidos como *input* por numa determinada máquina necessários para produzir 1 unidade da matéria-prima Z.

Encontrando o maior número possível destes padrões, o cliente irá ter que inserir um número substancialmente menor de *inputs* no cmNavigo. Assim que forem encontrados todos os padrões possíveis, a próxima fase seria integrar o cmNavigo com o simulador escolhido.

Já no segundo caso o projeto cresce e fica ao mesmo tempo mais interessante do ponto de vista científico e pedagógico como do ponto de vista da CM. Caso não exista nenhum *software*

compatível com o cmNavigo e seja necessário efetivamente desenhar um de raiz, o próprio cmNavigo irá valorizar-se, pois como já foi explicado a simulação é uma mais-valia para qualquer produto que trabalhe com os sistemas de uma empresa e este ganharia assim um simulador personalizado. A pesquisa passaria não pela pesquisa da estrutura das diferentes linhas de produção mas sim pela conceção e desenvolvimento de um modelo de simulação e de simulador para um MES. Esta pesquisa envolveria o processo de ter como *input* o fluxo de produção do sistema do cliente, assim como a personalização do modelo de simulação tendo em conta estes inputs, e transformá-los numa máquina de estados, criando assim o modelo conceptual onde irá ocorrer a simulação.

A solução final irá ser criada num ambiente de desenvolvimento fornecido pela CM, alojado num dos seus servidores. Aqui irá ser possível aceder a dados de teste antes de ser efetuada a transição para dados reais fora desse mesmo ambiente. Irá ser possível validar a solução quando esta conseguir simular com precisão as operações do cmNavigo numa linha de produção de um cliente e esta simulação se assemelhar ao que acontece na realidade na interação entre a ferramenta e a linha de produção efetivamente existente.

1.6 Contribuições Esperadas

A grande inovação deste projeto está relacionada com a conceção e desenvolvimento de uma solução de criação de ambientes e simulação, com dados específicos de cliente, sobre um MES já existente, ou seja, a conceção de um modelo inovacor e o desenvolvimento do simulador. Existem diversas soluções empresariais de simulação de operação fabril, mas são pesadas e de difícil utilização. Esta será uma solução leve, que permitirá com enorme rapidez dar uma ideia concreta de como se portará o sistema quando aplicado numa determinada realidade específica.

1.7 Estrutura do Relatório

Este relatório está dividido em oito capítulos.

No primeiro é feita a introdução do tema, a sua contextualização, motivação e principais objetivos, assim como a formalização do problema e a metodologia usada para o resolver. No segundo é feita a revisão da literatura de todos os conhecimentos que são necessários à realização deste projeto. No terceiro capítulo é efetuado o levantamento do estado da arte dos trabalhos relacionados com este projeto e das tecnologias usadas. No quarto será apresentada em detalhe a abordagem metodológica usada na resolução do projeto. No quinto capítulo estará descrito em detalhe o desenvolvimento e a implementação do protótipo do simulador. No sexto capítulo serão apresentadas os principais resultados que foram possíveis retirar depois de concluído o protótipo. No sétimo são apresentados os passos que foram efetuados para validar e

verificar o protótipo junto da CM. Por fim, no capítulo oito, são apresentadas as principais conclusões retiradas e o trabalho futuro proposto.

Capítulo 2

Revisão Bibliográfica: Background

Neste capítulo são descritos os conceitos inerentes ao tema do projeto. É feito um estudo sobre a simulação, definindo-a e fazendo um levantamento dos principais modelos e paradigmas existentes. Numa segunda parte é apresentado o estudo sobre os *Manufacturing Execution System*, começando por definir o conceito, explicar as principais tecnologias e arquiteturas existentes, enumerar as principais limitações que existem atualmente neste tipo de produtos e é ainda explicado o futuro desta tecnologia. Por fim, é feito um estudo da ferramenta que irá ser usada, o cmNavigo.

2.1 Simulação

A simulação é um termo bastante complexo, com muitas *nuances*, mas que se pode definir recorrendo ao dicionário como "*a técnica de imitar o comportamento de alguma situação ou sistema através de um modelo ou aparelho semelhante, para obter informações de forma mais convicente ou para treinar pessoal.*". No âmbito deste projeto, a simulação em causa é a Simulação Computorizada, e por isso o modelo acima mencionado é executado num computador. Fishwick conclui que a Simulação Computorizada é "*a disciplina de conceção de um modelo de um sistema físico real ou teórico, executando-o num computador e analisando os resultados.*" [FPA95]. De modo a perceber como este tipo de simulação é realizado, foi efetuado um estudo detalhado sobre a simulação e a conceção de modelos.

2.1.1 Definição/Conceitos

A simulação computorizada surgiu em meados de 1940 graças a um grande avanço tecnológico na altura: a invenção do primeiro computador acessível aos investigadores e que era

capaz de ser reprogramado para resolver problemas matemáticos - o ENIAC. Através deste computador, foi possível utilizar o Método de Monte Carlo em computadores eletrónicos, a fim de resolver certos problemas de difusão de neutrões que surgiram no projeto da bomba de hidrogênio e que eram (e ainda são) analiticamente intratáveis. [GOD10] Surgiu assim a simulação computorizada dos tempos modernos.

Segundo Shannon, a simulação é "*o processo de conceção de um modelo de um sistema real e a realização de experiências com este modelo para o propósito de compreender o comportamento do sistema e/ou avaliação de várias estratégias para o funcionamento do sistema*" [SRE98] Já Carson constata que "A simulação é uma ferramenta poderosa para a avaliação e análise de projetos de novos sistemas, modificações em sistemas já existentes e às alterações propostas para os sistemas de controlo e regras de funcionamento. Conduzir uma simulação válida é ao mesmo tempo uma arte e uma ciência." [CJS05] Consegue-se constatar então que ambos concordam que a simulação é, tal como Goldsman simplifica, "uma imitação da operação de um sistema do mundo real para fins de avaliação desse sistema." [GOD07]

Mas não chega definir simulação ou simulação computorizada, é preciso definir conceitos como "sistema" ou "modelo" que estão interligados com a própria definição de simulação. [PAP09]

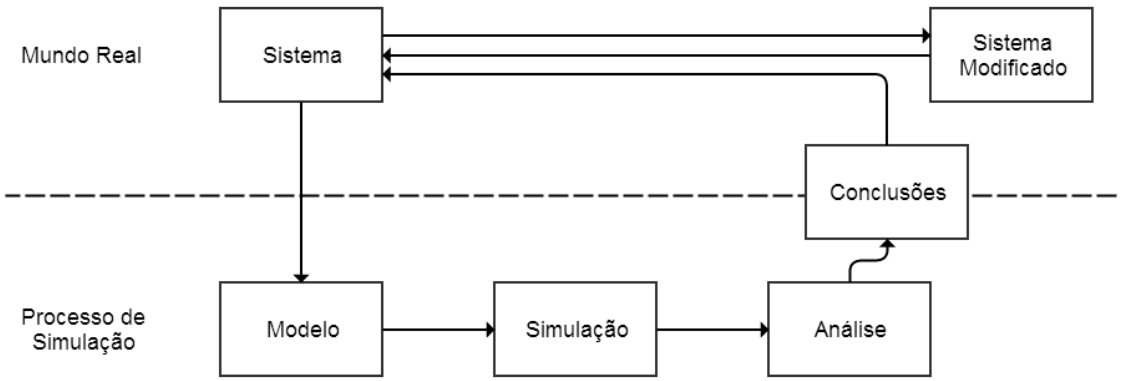


Figura 2: Esquema de um Estudo de Simulação

2.1.1.1 Sistema

Um sistema, no âmbito da simulação, é um conjunto de elementos interconectados que formam um todo organizado e que vão ser simulados. É a partir do sistema que é criado o modelo da simulação. [MAA97]

É possível ver na Figura 2 a relação que o sistema tem com o processo de simulação. Existe um sistema real no mundo real que é o sistema em análise. Esse mesmo sistema é transformado num modelo, que irá ser simulado e analisado de maneira a retirarem-se as

conclusões que estão em estudo. Com essa informação, o sistema é então modificado e melhorado, transformando-se num novo sistema, melhor que o anterior. Este processo é repetido, sustentando assim a melhoria contínua do sistema. [MAA97]

2.1.1.2 Modelo

A modelação é o processo de produzir um modelo, que por sua vez é a representação da construção e funcionamento do sistema em estudo [MAA97] [CJS05]. Um modelo de simulação é a representação que incorpora o tempo e as mudanças que ocorrem durante esse intervalo. [CJS05]

Um modelo tem que ser ao mesmo tempo o mais semelhante possível ao sistema real, incorporando as suas características principais, e ao mesmo tempo não pode ser tão complexo quanto este senão seria impossível desenhá-lo e analisá-lo. [MAA97]

Segundo Maria, o desenvolvimento de um modelo de simulação está dividido em 6 passos:

- **Identificar o problema**, enumerando os problemas existentes no sistema a ser estudado;
- **Formular o problema**, definindo o objetivo geral do estudo e algumas questões específicas a serem abordadas;
- **Recolher e analisar os dados de um sistema**, em concreto os dados sobre as especificações do sistema, variáveis de entrada, bem como o desempenho do sistema existente;
- **Formular e desenvolver um modelo**, desenvolvendo esquemas e diagramas de rede do sistema;
- **Validar esse modelo**, comparando o desempenho do modelo em condições conhecidas com o desempenho do sistema real;
- **Documentar o modelo para futuros utilizadores**, como por exemplo objetivos, teorias e variáveis de entrada em detalhe;

2.1.2 Modelos

Os modelos de simulação podem ser categorizados de acordo com vários pares independentes de atributos: [PAP09]

- Estocástico ou Determinista (que inclui o modelo caótico);
- Estático ou Dinâmico;
- Contínuo ou Discreto (que inclui os eventos discretos ou modelos ED);
- Local ou Distribuído.

2.1.2.1 Estocástico vs. Determinista

Padrões estocásticos são aqueles que têm origem em processos não determinísticos, com origem em eventos aleatórios. Aqui os valores introduzidos na simulação são aleatórios. [AIG77]

Um sistema determinista é um sistema no qual nenhuma aleatoriedade está envolvida no desenvolvimento de estados futuros do sistema. Um modelo determinístico produz assim sempre o mesmo resultado de uma determinada condição de partida ou estado inicial. Aqui os valores introduzidos na simulação são constantes. [SRR02]

2.1.2.2 Estático vs. Dinâmico

Os modelos estáticos representam estruturas que não dependem do tempo. Isto inclui a modelação de estruturas organizacionais, dos portadores de informação, como formulários, ou a modelagem de relações entre objetos de negócios. O estado do sistema é apenas descrito num determinado momento no tempo. Aqui o tempo não é uma variável necessária. [SAP14]

Sempre que um modelo é suposto mostrar informação relevante sobre um processo que pode ser colocado de uma forma dependente do tempo cronológico, este tipo de modelo é referido como dinâmico. Aqui o estado do sistema é descrito com base na variável de tempo, fazendo com que o sistema evolua ao longo deste. [BRN98]

2.1.2.3 Contínuo vs. Discreto

Modelos contínuos são modelos onde a variação envolve transições graduais e quantitativas, sem mudanças abruptas ou descontinuidades. Neste caso o tempo da simulação progride de forma contínua em intervalos de tempo iguais e definidos. [GGC86]

Um modelo discreto é um tipo de modelos onde a simulação é controlada por eventos, ou seja, só avança de uma atividade para outra quando um evento é despoletado. [GGC86]

2.1.2.4 Local vs. Distribuído

Num sistema local, a simulação é processada e calculada apenas numa máquina sem ser necessário recorrer a fontes externas.

Um sistema distribuído é um sistema de *software* onde os componentes estão localizados em computadores ligados em rede, comunicam e coordenam as suas ações através de mensagens. Os componentes interagem uns com os outros, a fim de atingir um objetivo comum.

No caso do projeto a ser desenvolvido, pode-se desde já classificar o modelo como determinista, já que não haverá nenhuma aleatoriedade nos dados de entrada, dinâmico, já que

evoluirá ao longo do tempo, será contínuo e discreto, pois incluirá as duas variações em alturas e atividades diferentes da simulação, e local.

2.1.3 Paradigmas

Um paradigma é um conceito das ciências e da epistemologia que define um exemplo típico ou modelo de algo. É a representação de um padrão a ser seguido. De seguida são apresentados os principais paradigmas da simulação atual. [HBP12]

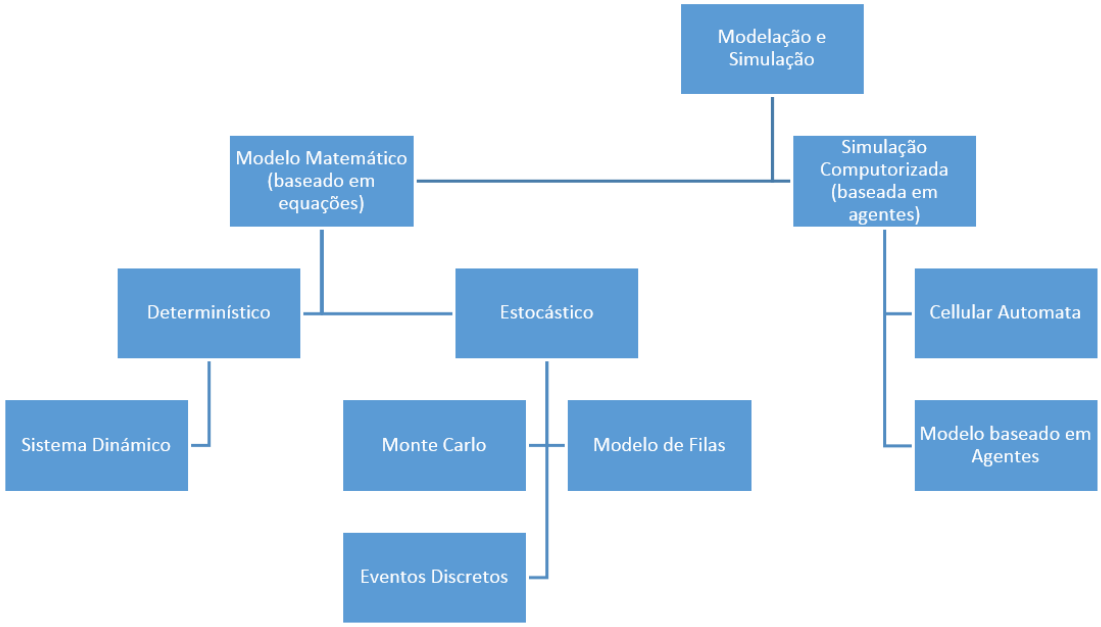


Figura 3: Principais Paradgimas da Simulação existentes

Podemos alocar os vários paradigmas da simulação em relação à forma como é feita a simulação em dois grandes grupos: do geral para o particular, que são todos os paradigmas do Modelo Matemático, e do particular para o geral, os paradigmas da Simulação Computorizada. O Modelo Matemático não é mais que um modelo baseado em equações, enquanto a Simulação Computorizada é baseada em agentes. É um grupo de paradigmas relativamente novo, já que resulta de pesquisas recentes no campo da inteligência artificial. [HBP12]

Tanto o Sistema Dinâmico como o Modelo de Eventos Discretos são exemplos típicos de abordagens baseadas em equações, pois usam essas mesmas equações para determinar o próximo estado do sistema. O primeiro é determinístico, ou seja, comporta-se sempre da mesma maneira. Caso lhe sejam passados os mesmos dados várias vezes, o resultado será sempre o mesmo. Já os segundos (Modelo de Monte Carlo, Modelo de Filas e Modelo de Eventos Discretos) são baseados em processos estocásticos, já que incluem equações diferenciais

estocásticas, equações que envolvem números gerados aleatoriamente para simular incerteza. [HBP12]

Por outro lado, a Simulação Computorizada assume a perspetiva do agente, em contraste com a perspetiva baseada em processos dos modelos matemáticos. Por isso é que os Modelos Matemáticos têm uma abordagem *top-down* (geral-particular) enquanto esta é *bottom-up* (particular-geral). Os modelos baseados em agentes são processados de acordo com as interações entre cada unidade individual, entre cada agente, tendo cada um deles regras de inteligência e comportamento próprias. [HBP12]

2.1.3.1 Sistema Dinâmico

Segundo o Professor Jay Forrester, o criador deste modelo, os Sistemas Dinâmicos são "*o estudo da informação/feedback das características da atividade industrial para mostrar como as estruturas organizacionais, as amplificações (em políticas) e os atrasos temporários (em decisões e ações) interagem para influenciar o sucesso de um empreendimento* " [BAF04]. Por outras palavras, os Sistemas Dinâmicos são sistemas que tratam os loops de feedback internos e atrasos que afetam o comportamento de todo o sistema. É uma das abordagens para a compreensão do comportamento de sistemas complexos ao longo do tempo que requer um conhecimento bastante amplo sobre como as variáveis de estado do sistema interagem entre si, podendo-se concluir que é um modelo muito abstrato. Neste tipo de modelação, os sistemas do mundo real e os seus processos são representados em termos de *stocks* (material, conhecimento,

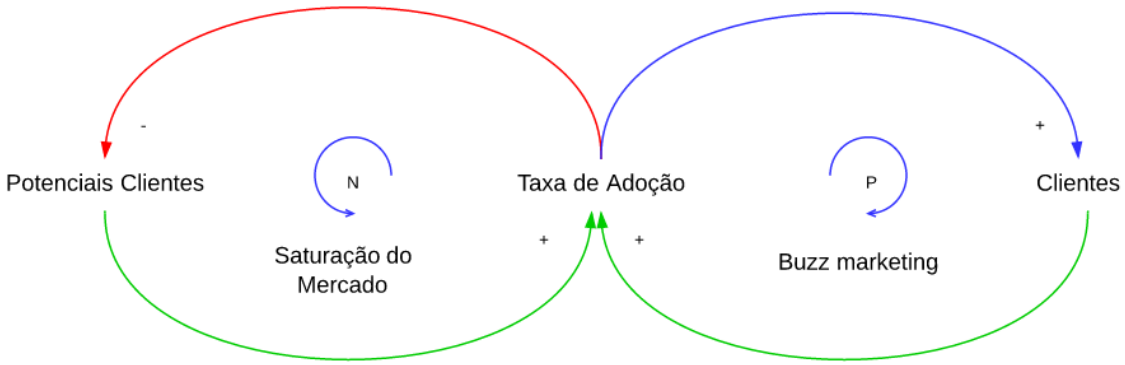


Figura 4: Diagrama de Laço Causal

pessoas), fluxos entre essas stocks e informação que determina o valor desses fluxos.

O número de aplicações que este modelo abrange é elevado, indo de sistemas urbanos e sociais à modelação de sistemas ecológicos, usada tanto no setor privado como no público. Usando a simulação é possível estimar a evolução de variáveis e indicadores críticos, tal como o sucesso da venda de um novo produto no setor privado ou a poluição, a pobreza ou o

desemprego no setor público. É uma técnica eficaz para o enquadramento, compreensão e discussão de problemas difíceis e opções políticas para os enfrentar. [SJD03] [BAF04]

A aplicação de um Sistema Dinâmico a um problema concreto mais largamente discutido e explicado é precisamente o sucesso da venda de um novo produto. O primeiro passo para a conceção deste modelo passa pelo desenho de um Diagrama de Laço Causal. Um DLC não é mais que um diagrama que ajuda a visualizar como as diferentes variáveis de um sistema estão relacionadas. A Figura 4 é o DLC do problema mencionado. [SJD03] [BAF04]

Existem dois loops de feedback neste diagrama. O reforço positivo P (à direita) indica que quanto mais pessoas adotarem o novo produto, mais forte será o Buzz marketing, mais referências para o produto irão existir. O reforço negativo N (à esquerda), ou balanço do mercado, indica não poderá haver crescimento para sempre pois quanto mais pessoas adotarem o novo produto, menos potenciais clientes haverão. As setas verdes indicam que a Taxa de Adoção é uma função de Potenciais Clientes e Clientes, a seta vermelha indica que existe um declínio dos Potenciais Clientes com o crescimento da Taxa de Adoção e no sentido contrário, a seta azul indica que os Clientes aumentam. [BAF04]

Para poder ser feita uma análise mais quantitativa do esquema, não chega um DLC. Introduzem-se então as ações e os fluxos para formar um novo diagrama. Neste exemplo existem duas ações, os Potenciais Clientes e os Clientes, e existe um fluxo, os Novos Clientes. Para cada Potencial Cliente perdido, é ganho um novo Cliente. Com estes dados é possível criar então um conjunto de equações diferenciais que vão formar o modelo matemático que se está à procura, como por exemplo as duas equações mais fáceis de se se obter do diagrama. [SJD03]

$$Potenciais\ Clientes = -\int_0^t Novos\ Clientes\ dt$$
$$Clientes = \int_0^t Novos\ Clientes\ dt$$

2.1.3.2 Modelo de Monte Carlo

O Modelo de Monte Carlo é usado principalmente na modelagem de sistemas com bastante incerteza nos dados de entrada e onde queremos calcular, por exemplo, a média e a distribuição da produção. Algumas das aplicações da simulação que usam este método incluem sistemas nas áreas da engenharia, telecomunicações e finanças. [CHY11]

A forma de execução deste modelo é emparelhar cada dado de entrada com um número aleatório apropriado, isto é, com uma probabilidade de distribuição apropriada e parâmetros relevantes. É então produzido um primeiro conjunto de valores e é com base neles que os primeiros dados de saída são calculados. Este processo é repetido um grande número de vezes, o que possibilita o cálculo da média dos valores aleatórios, assim como a distribuição dos dados de saída. [CHY11]

2.1.3.3 Modelo de Filas

A teoria das filas é o estudo matemático das filas de espera onde se tenta prever o tamanho da fila, assim como os tempos de espera, através da conceção de um modelo. A teoria foi concebida inicialmente em 1909 por Agner Erlang, mas em 1953 David Kendall introduziu a notação que hoje em dia é usada nos modelos matemáticos: G/G/n. Com este modelo é possível calcular várias medidas de desempenho, como é o caso do tempo médio de espera numa fila, o número previsto de elementos em espera ou a receberem um serviço e a probabilidade do sistema estar num determinado estado. [SAM08]

Caso se consiga modelar o problema para um Modelo de Filas, a grande vantagem de usar este modelo é a rapidez dos cálculos dos resultados, pois se a solução analítica já está calculada (a modelação do problema), então só resta calcular as fórmulas a serem usadas e obter o resultado. A principal limitação desta teoria é por vezes ser demasiado matemática e abstrata e não poder ser modelada para problemas do mundo real. Por exemplo, os modelos matemáticos podem assumir um número infinito número de clientes ou um número infinito de lugares disponíveis na fila para conseguirem efetuar os cálculos. Pode-se concluir que este tipo de modelos pode apenas ser usado para resolver estimativas de sistemas do mundo real superficiais. [SAM08] [HBP12]

2.1.3.4 Modelo de Eventos Discretos

A grande diferença deste paradigma para os restantes é a forma como o tempo é tratado. Nos modelos de Eventos Discretos o tempo não é contínuo, não avança em passos iguais. Só avança quando existe um evento que muda algo no estado atual, o que permite que se possa avançar para o estado seguinte. Por isso pode-se concluir que o tempo só avança praticamente de evento para evento. [CHY11]

A simulação de Eventos Discretos envolve a modelação do sistema organizacional como um conjunto de entidades que evolui ao longo do tempo de acordo com a disponibilidade de recursos e o desencadeamento de eventos. Existem mais benefícios no uso deste tipo de modelos quando o sistema que se pretende estudar pode ser caracterizado por variáveis, pelos estados correspondentes e há a ocorrência de eventos que alteram os valores desses estados variáveis. Não é apropriado o uso deste tipo de simulação quando as variáveis de estado interagem entre si de uma forma quase contínua ou quando as entidades e os seus mecanismos internos são mais importantes que a simulação dos eventos por si só. [HBP12]

Embora existam diversas variações deste paradigma, todos evoluíram da mesma estrutura básica inicial. Independentemente de quão complexas possam ser, é seguro dizer que todas essas variações têm os componentes básicos que de seguida são descritos.

i. Entidade

Uma entidade é um objeto do modelo que causa algum tipo de mudança no estado da simulação [CJS05] [IRG08]. As entidades têm ainda atributos que representam as características dessa entidade e que lhe são inerentes. Estes atributos são essenciais na análise da simulação, pois é através deles que são gerados os dados necessários para o estudo do modelo em causa.

Uma entidade especial são os recursos. Este tipo de entidade presta um serviço às entidades normais e é também definida por ter uma capacidade limitada. [IRG08].

ii. Evento e Atividade

As atividades são os processos e a lógica na simulação. Os eventos são condições que ocorrem num determinado ponto no tempo e que provocam uma mudança no estado do sistema. Uma entidade interage com as atividades, criando eventos. [IRG08] Tendo como exemplo comprar um bolo, as entidades são o sujeito que compra o bolo e o sujeito que vende, um evento é pagar o bolo e duas atividades são escolher o bolo e receber o bolo que foi comprado.

Podemos ainda caracterizar as atividades e os eventos em relação à forma como são programados. Os eventos podem-se dividir em:

- Eventos primários, gerados por dados; [CJS05]
- Eventos secundários, gerados internamente pelo modelo lógico. [CJS05]

Já as atividades podem-se caracterizar como:

- Atrasos, quando as entidades são atrasados um período de tempo definido; [IRG08]
- Filas, quando as entidades esperam um período de tempo indefinido; [IRG08]
- Lógicas, quando permitem que uma entidade afete o sistema. [IRG08]

iii. Variáveis Globais

Uma variável global é uma variável que está disponível para todo o modelo em qualquer momento da simulação. Uma variável global pode controlar praticamente qualquer coisa que é de interesse de toda a simulação. [IRG08]

iv. Gerador de Números Aleatórios

Um gerador de números aleatórios é tecnicamente chamado de pseudo-gerador de números aleatórios pois os computadores não têm capacidade de gerar aleatoriedade. Este gerador não é mais que uma rotina que retorna um número entre 0 e 1, número esse que é usado nas distribuições aleatórias que acontecem na simulação. [IRG08]

v. Calendários

O calendário não é mais que a lista dos eventos que estão programados para acontecer. Em cada simulação existe um calendário de eventos que está ordenado do primeiro evento a ocorrer. [IRG08]

vi. Variáveis de Estado do Sistema

Podem existir várias variáveis de estado para os diferentes sistemas a serem simulados, mas uma que é recorrente em qualquer simulação é o tempo atual da simulação. É feito a actualização desta variável sempre que uma entidade é retirada do calendário. [IRG08]

vii. Colecionadores de Estatísticas

Por fim, os colecionadores de estatísticas servem para obter o resultado de algumas das variáveis da simulação, como por exemplo o número de recursos gastos num determinado estado. No fim, a validação de um modelo só é possível após a análise das estatísticas que daqui advêm. [IRG08]

2.1.3.5 Cellular Automata

Um autómato celular é um modelo discreto focado em vários agentes, caracterizados pelo comportamento à base de regras pré-definidas e que têm interações locais com o espaço físico. É constituído por uma coleção de células (agentes) que podem ter diferentes cores, sendo que cada cor representa um estado possível diferente, e estão ordenadas numa grelha que pode ter diferentes formatos. O estado de cada célula é alterado tendo em conta o estado das células que lhe são adjacentes. Normalmente cada célula é binária, tem apenas dois estados, no entanto é possível conceber autómatos com vários estados para resolver problemas mais complexos. As regras internas de cada célula são aplicadas iterativamente tantas vezes quantas as que forem necessárias e as regras de transição, que são os possíveis estados de cada célula e por isso os principais componentes do modelo, são geralmente comuns a todas as células, daí estes modelos serem sistemas homogéneos. A abordagem efetuado conta com a divisão do modelo em sub-modelos, onde cada diferente conjunto de células representa uma parte de todo o sistema. [CHY11] [BMD98]

O autómato celular mais simples é unidimensional. Cada célula é binária e tem como vizinhos apenas a célula que a antecede e a que a sucede. Assim, uma célula e as suas duas vizinhas formam uma vizinhança de 3 células, por isso existem $2^3=8$ padrões possíveis para uma vizinhança. Há então $2^8=256$ regras de transição possíveis para esse modelo. O autómato celular bidimensional mais conhecido é o famoso jogo da vida, que é um autómato que simula processos de evolução de células biológicas. [BRA99]

2.1.3.6 Modelo Baseado em Agentes

A Simulação Baseada em Agentes (ABS) é uma nova abordagem para os sistemas de simulação com a interação de agentes autónomos, como os exemplos descritos em [RLT11]. Enquanto a definição precisa da ABS varia entre as diferentes áreas de aplicação (ou até mesmo na mesma área), a filosofia e os usos deste paradigma são semelhantes, simulando interações de objetos autônomos (chamados agentes) para identificar, explicar, gerar e projetar comportamentos emergentes. [WKV10]

Os principais passos nesta simulação são a construção de vários agentes que têm regras e inteligência próprias, o cálculo/simulação das interações entre esses agentes e o estudo do macro-sistema que daí emerge. Um agente pode ser definido por um programa independente que controla as suas próprias ações com base no comportamento que lhe foi incutido e na perceção que tem do ambiente operacional em que está inserido, ambiente este que tem por nome vizinhança e é constituído pelos agentes que o rodeam. O comportamento chave que um componente tem que ter para ser considerado um agente é a capacidade de se adaptar à sua vizinhança através da interação e observação da mesma. Para isso as regras definidas para o agente devem ter dois componentes: as regras de baixo-nível que regem o comportamento do agente e as regras de alto-nível que permitem a adaptação e evolução do comportamento do agente com base na experiência que retiram da interação com o mundo que o rodeia. [HBP12]

Um Modelo Baseado em Agentes é caracterizado por cinco atributos: [CHY11]

- **Heterogeneidade**, que se refere ao facto dos agentes alterarem o seu comportamento inicial durante a simulação levando a resultados inesperados, e mais próximos do mundo real, durante a simulação; [CHY11]
- **Autonomia**, que é a capacidade do agente tomar as suas próprias decisões, e não se limitar apenas a seguir as indicações e regras de um sistema superior, exatamente como é feito numa abordagem *top-down*; [CHY11]
- **Espaço explícito** é o mundo onde é feita a simulação, normalmente composto de um autómato celular de n dimensões; [CHY11]
- **Interações locais**, pois um agente apenas interage com os agentes que fazem parte da sua vizinhança, e não com todos que constituem o sistema; [CHY11]
- **Racionalidade limitada**, que pode ser separada em dois termos diferentes: informação limitada e capacidade de processamento limitada. A primeira diz-nos que o agente segue determinadas regras com base no conhecimento local, ignorando o conhecimento geral do sistema e a segunda limita a capacidade de processamento do agente, de maneira a este não conseguir efetuar todas as tarefas sozinho. [CHY11]

A metodologia MABLe (Modelação do Comportamento dos Agentes por Aprendizagem) é baseada na ideia de que um modelo concetual do sistema pode ser criado através da divisão das regras dos agentes em duas categorias, sensores e efetores. No primeiro grupo estão incluídos

todos os comportamentos que permitem ao agente recolher informação do ambiente, através das suas perceções, e no segundo grupo estão as regras de atuação do agente ou ações. Em particular, os dez passos para um uso correto deste tipo de metodologia são: [JRF12]

1. **Separar os agentes e o ambiente**, determinando quais são as as metas a atingir com a simulação;
2. **Identificar os aspetos relevantes**, como dinâmicas globais e entidades locais;
3. **Determinar as ações primitivas do agente** (ou efetores) no ambiente e como este muda com essas ações;
4. **Determinar que informação do ambiente é passada ao agente** (via sensores);
5. **Decidir a arquitetura de aprendizagem**, que é capaz de conectar as percepções e ações do agente de forma adequada para este efetuar realmente uma aprendizagem;
6. **Determinar a função de cálculo da performance de cada agente**. Este passo é de extrema importância pois é através da performance individual de cada agente que os outros agentes conseguem perceber se o comportamento desse agente é algo a seguir ou a evitar.
7. **Implementar o modelo do ambiente**, incluindo a já mencionado função de cálculo da performance;
8. **Especificar e implementar os agentes**, o seu comportamento básico, interações e o respetivo mecanismo de aprendizagem escolhido;
9. **Simular, verificando a aprendizagem dos agentes**, identificando potenciais situações que possam ocorrer devido a um modelo do ambiente impróprio, a um fraco cálculo da performance ou a uma má interação entre os agentes;
10. **Analisar os resultados obtidos**, análise esta que pode ser feita comparando o comportamento aprendido dos agentes com alguma hipótese existente ou com uma solução já existente.

Este é um processo iterativo que deverá ser repetido até os resultados serem aceitáveis, tendo no fim um modelo ds agentes que tiveram uma boa performance no ambiente de simulação estudado. [JRF12]

Na Tabela 1 é possível ver um resumo das principais características de cada paradigma. [HBP12]

Tabela 1: Características dos principais Paradigmas

Metodologia	Tempo	Processo	Nível	Tipo
Sistema Dinámico	Contínuo	Determinístico	Macro	Top-Down

Monte Carlo		Estocástico	Micro	
Método das Filas	Discreto		Micro-Meso	
Eventos Discretos				
Cellular Automata		Determinístico	Meso-Macro	Bottom-Up
Baseado em Agentes				

2.2 Manufacturing Execution System

O setor industrial sempre foi o grande motor do avanço tecnológico desde o *boom* da revolução industrial. A automação industrial utiliza a mesma infraestrutura tecnológica há décadas sem precisar de sofrer alterações de renome pois se o processo funciona bem, não há razão para ser alterado. Já a gestão da indústria é uma área bem mais recente que utiliza tecnologias mais atuais. A infraestrutura é composta por computadores ligados localmente e o armazenamento da informação é feito em bases de dados. [KIT12] [ELR13]

De início surgiram vários programas que permitiam a gestão industrial mas com o passar do tempo foram todos incorporados num grande módulo: os Enterprise Resource Planning (ERP). Estes sistemas de informação têm toda a informação de todos os estados do negócio, nomeadamento sobre:

- Planeamento do produto e dos custos de produção;
- Marketing e vendas;
- Inventório;
- Envios e pagamentos.

Esta gestão, e sobretudo o planeamento, é feita não considerando qualquer tipo de restrições na linha de produção. Para os ERP, a produção é uma misturadora onde se coloca matéria-prima, recursos humanos e os custos envolvidos e obtém-se produtos acabados num determinado espaço de tempo. Os ERP ajudam sim a produção, mas mais a um nível administrativo, não se preocupando com a informação detalhada do que ocorre na produção, quais os problemas mais recorrentes e como os resolver, onde investir para melhorar o fluxo da produção e quando e quem treinar. E foi devido a todas estas falhas que surgiram os Manufacturing Execution Systems. [KIT12]

2.2.1 Definição e Funcionalidades

A organização MESA define formalmente um MES como "*Um Manufacturing Execution System recolhe informação que permite a otimização das atividades de produção desde a ordem de produção até aos produtos acabados. Usando dados atuais e precisos, um MES guia, inicia, responde a e cria relatórios sobre as atividades da fábrica à medida que ocorrem. A rápida resposta às constantes mudanças, juntamente com um foco na redução de atividades que não acrescentam valor ao produto final, torna possível dirigir as operações e os processos da fábrica eficazemente. O MES melhora o retorno sobre os ativos operacionais, bem como as entregas dentro do prazo, a rotação do* stock*, a margem bruta e o desempenho do fluxo de caixa. Um MES fornece informações críticas sobre as atividades de produção em toda a empresa e da cadeia logística através de comunicações bidirecionais.*" [SUB09]

Podemos definir então um MES como uma ponte entre os sistemas de informação de mais alto nível, os ERP, e os sistemas que controlam os processos ao nível da oficina. O seu principal objetivo é verificar se o que foi planeado no nível mais alto está a ser corretamente executado

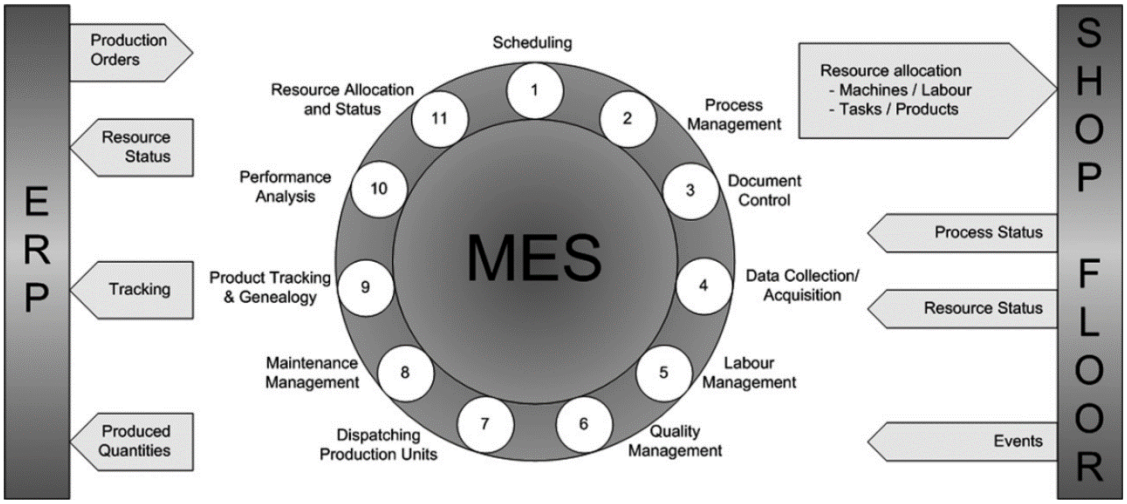


Figura 5: Funcionalidades de um Manufacturing Execution System

no nível inferior. O seu propósito é automatizar ainda mais a indústria, reduzindo os desperdícios e aumentando o tempo de atividade das máquinas, traduzindo-se isto no fim numa melhoria geral do desempenho da linha de produção.

A mesma organização definiu ainda as principais funcionalidades de um MES, que estão sumarizadas na Figura 5. [KLJ07] [SUB09]

- **Planeamento das Operações**: Sequência e tempos das atividades para obter a melhor performance possível da linha de produção, tendo em conta os recursos disponíveis.
- **Gestão dos Processos**: Direcionar o fluxo de trabalho na linha de produção com base no planeamento feito das atividades de produção reais.
- **Controlo de Documentos**: Gerir e distribuir informação sobre os produtos, processos e ordens, bem como reunir declarações de certificação de trabalho e condições.

- **Recolha de Dados**: Monitorização, recolha e organização de dados sobre processos, materiais e operações de pessoas, máquinas ou controlos.
- **Gestão de Trabalho**: Monitorização e orientação da utilização de pessoal durante um turno com base nas qualificações, padrões de trabalho e necessidades comerciais.
- **Gestão de Qualidade**: Gravação, monitorização e análise de características de produtos e processos contra os ideais de engenharia.
- **Despacho das Unidades de Produção**: Dar o comando para enviar materiais ou ordens para certas partes da planta para começar um passo ou um processo.
- **Gestão de Manutenção**: Planear e executar as atividades adequadas à correta manutenção do equipamento ou de outros bens.
- **Monitorização e Geneologia do Produto**: Monitorizar o progresso de unidades, lotes ou lotes de produção para criar uma história completa do produto.
- **Análise de Desempenho**: Comparar os resultados medidos na linha de produção com metas e métricas definidas pela corporação, os clientes ou os órgãos reguladores.
- **Alocação e Estado dos Recursos**: Orientar o que pessoas, máquinas, ferramentas e materiais devem fazer e ao mesmo tempo acompanhar o que eles estão a fazer atualmente ou que tenham acabado de fazer.

2.2.2 Tecnologias

Os produtos desenvolvidos nesta área são normalmente divididos em módulos, de maneira a que o cliente possa usar apenas os módulos que necessita para a sua empresa. Contudo, todos eles se podem dividir em três camadas principais: [SUB09]

- **Aplicação do Cliente/Servidor**: É a camada visível para o utilizador. Aqui estão agrupados todos os módulos que permitem interação com a aplicação, assim como é nesta camada que outros sistemas podem ser "ligados", permitindo a sua comunicação.
- **Framework de Integração**: Esta camada é o centro de toda a arquitetura do programa. É constituída por objetos padrão reutilizáveis que juntos formam a infraestrutura da aplicação, por onde o fluxo da informação passa.
- **Armazenamento/Gestão de Dados**: Esta camada fornece serviços essenciais às duas camadas anteriores, tal como comunicação entre redes e serviços de gestão de objetos. Estes serviços devem ser construídos de uma forma robusta e independente de sistemas operativos e tecnologias de base de dados para fornecer uma base sólida e de longo prazo a todo o sistema.

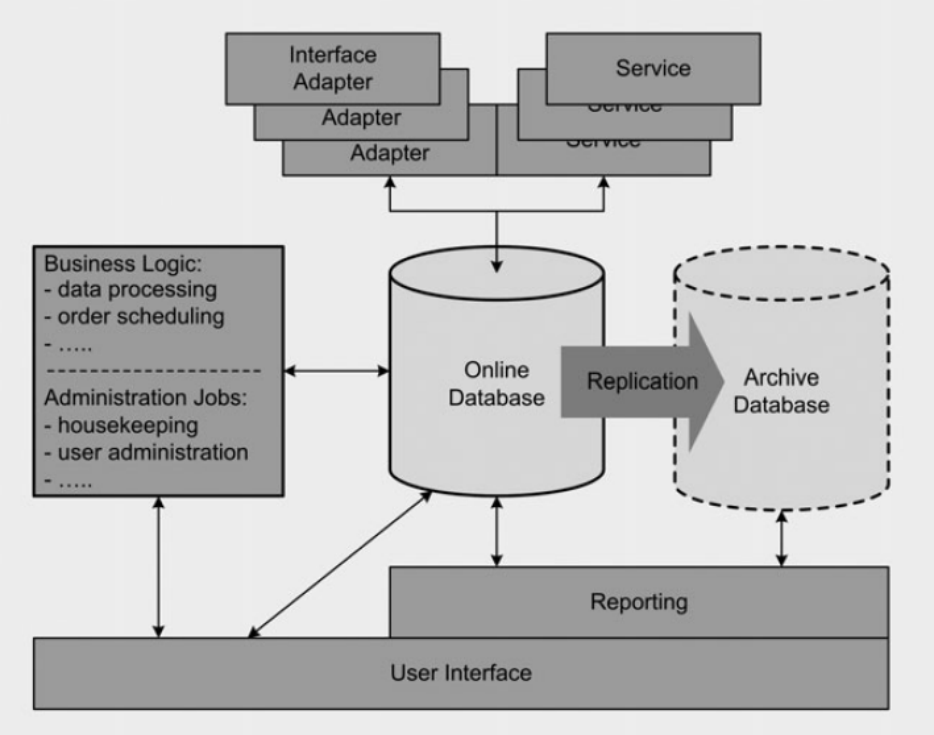


Figura 6: Arquitetura básica de um MES

2.2.3 Arquitetura

Na Figura 6 é possível ver um sistema baseado num servidor, cujo núcleo é uma base de dados relacional. De modo a ser capaz de lidar com grandes conjuntos de dados, essa base de dados pode ser dividida em duas partes: uma base de dados *online* e uma base de dados de arquivo, sendo que na primeira está colocada normalmente toda a informação necessária para a execução do programa enquanto na segunda estão todos os dados de execuções anteriores. Algumas funções básicas de processamento de dados são também implementadas aqui mas as funções de processamento mais pesadas são tratadas num módulo externo. Neste módulo, por exemplo um servidor, a lógica de negócio e as tarefas administrativas são mapeadas. Por fim, existem adaptadores independentes da interface e/ou serviços de *software* que estão disponíveis como interfaces para a tecnologia de informação de sistemas vizinhos. A componente do servidor é também necessária para a interface do utilizador e para o módulo dos relatórios incorporado. [MEH09] [ELR13]

2.2.4 Atuais Limitações

As limitações presentes nos sistemas atuais de MES estão ligadas principalmente a três áreas: a arquitetura geral, o impacto da conectividade no processamento de dados e as dificuldades de integração deste tipo de sistemas.

2.2.4.1 Arquitetura

O principal problema arquitetural é a não existência de ferramentas para testar e validar novos avanços arquitetónicos em problemas realistas, tanto a nível do tamanho do sistema de manufatura como no rigor das técnicas de avaliação. É necessário a criação de um ambiente que permita testar estas mudanças em cenários de tamanho e complexidade reais em vez dos atuais testes.

2.2.4.2 Processamento em Tempo Real

Um MES tem de ser capaz de processar a informação do nível inferior (oficina) em tempo real, numa questão de horas ou até de minutos para conseguir controlar eficazmente as atividades que estão a decorrer. Mas quando os processos demoram segundos ou milissegundos, é necessário que o sistema não demore tanto tempo, senão perde-se a capacidade de processamento em tempo real.

Para reduzir o tempo, é preciso ligar o MES a mais equipamento para ele conseguir recolher mais dados para este saber mais cedo o que se está a passar na linha de produção e tentar responder mais rapidamente aos pedidos. Entramos então num ciclo vicioso e no verdadeiro problema: quanto mais perto um MES quer estar do processamento em tempo real, mais conexões tem que fazer; quantas mais conexões tem, mais dados recolhe; quantos mais dados recolher, menos perto está do processamento em tempo real pois mais dados significam mais tempo de processamento.

2.2.4.3 Integração

Quando se constrói um Manufacturing Execution System, o objetivo é que este seja possível de integrar nos sistemas de informação do máximo de clientes, por isso o *software* é desenhado o mais flexível possível. Em sentido contrário, os sistemas de informação dos clientes são feitos à medida de cada empresa. É então fácil de perceber que irão existir dificuldades na integração, apesar da flexibilidade implementada. [SUB09]

Uma das dificuldades mais facilmente percetíveis é em relação à hierarquia das tarefas de produção. O MES executa e controla as ordens que lhe chegam dos ERP, mas existem diferentes formas de produção que favorecem uma produção *bottom-up* como é o caso da produção JIT (Just In Time) ou da fabricação inversa. É então necessário efetuar mudanças no processamento das ordens, o que nem sempre é possível. [SUB09]

2.2.5 Futuro dos MES

De forma a resolver os problemas acima mencionados, é necessário redefinir os sistemas de MES tal como os conhecemos atualmente. [ELR13]

Uma das principais tendências emergentes tem por nome Reconfigurable Manufacturing Execution System. Tal como o nome indica, é um sistema onde os módulos podem ser facilmente reconfigurados ou trocados por novos módulos, invés de se substituir todo o sistema. Esta reconfiguração irá permitir adicionar, remover ou modificar processos específicos, sendo assim possível o sistema ajustar-se à capacidade de produção que se altera consoante a procura do mercado. [MMG00] [ELR13]

Uma outra abordagem tem por nome Holonic Manufacturing System que tem por base os Holons que constituem o sistema. Um Holon é algo que por si só é um todo e ao mesmo tempo é uma parte de um todo, por isso um Holon pode ser feito de outros Holons. Um Holon à semelhança de um agente é cooperativo e inteligente, o que possibilita que cumpram a sua função corretamente no todo em que estão integrados, mas ao mesmo tempo é autónomo. [SJM06] O exemplo mais simples que se consegue associar aos Holons são as formigas, que trabalham para o bem geral do formigueiro e cooperam entre si tomando decisões inteligentes mas ao mesmo tempo são indíviduos autónomos que conseguem tomar as suas próprias decisões. Existem muitas pesquisas sobre este tema na área da filosofia mas o que interessa retirar para esta explicação é a capacidade organizacional dos Holons (denominada de Holarchy) que permite construir sistemas muito complexos que são ao mesmo tempo eficientes e adaptativos a mudanças. [VPH00] [ELR13]

Por fim, uma terceira solução que adiciona flexibilidade aos MES atuais tem por designação Flexible Manufacturing System e que tenta que o sistema responda positivamente em caso de mudanças. Esta flexibilidade pode-se dividir em duas categoris: a flexibilidade das máquinas (capacidade de alterar as ordens de produção ou alterar todo o sistema para este produzir um novo produto) e a flexibilidade das rotas (capacidade do sistema absorver mudanças de grande escala, como mudanças na capacidade ou no volume de trabalho, e a possibilidade de várias máquinas efetuarem diferentes operações na mesma peça). [TAS98] [ELR13]

Existem várias vantagens da adoção destes novos métodos de construir um MES, podendo por exemplo ser reduzido os custos da manufatura de um produto, mais produtividade e eficiência das máquinas, assim como uma melhoria da capacidade do sistema como um todo, mas os custos que são precisos para haver uma nova implementação e a necessidade de pessoal especializado para conseguir trabalhar com estes sistemas mais complexos está a colocar um travão no possível avanço tecnológico. [MEH09]

2.3 cmNavigo

O cmNavigo é um Manufacturing Execution System produzido pela Critical Manufacturing e que é direcionado para as indústrias mais avançadas, como por exemplo a

indústria dos semi-condutores ou de dispositivos médicos. Este é o MES para o qual será produzido um simulador neste projeto. De seguida será apresentada a arquitetura geral do sistema, assim como os principais módulos que o constituem.

2.3.1 Arquitetura

Na Figura 7 é possível visualizar a arquitetura lógica do sistema que está dividido em

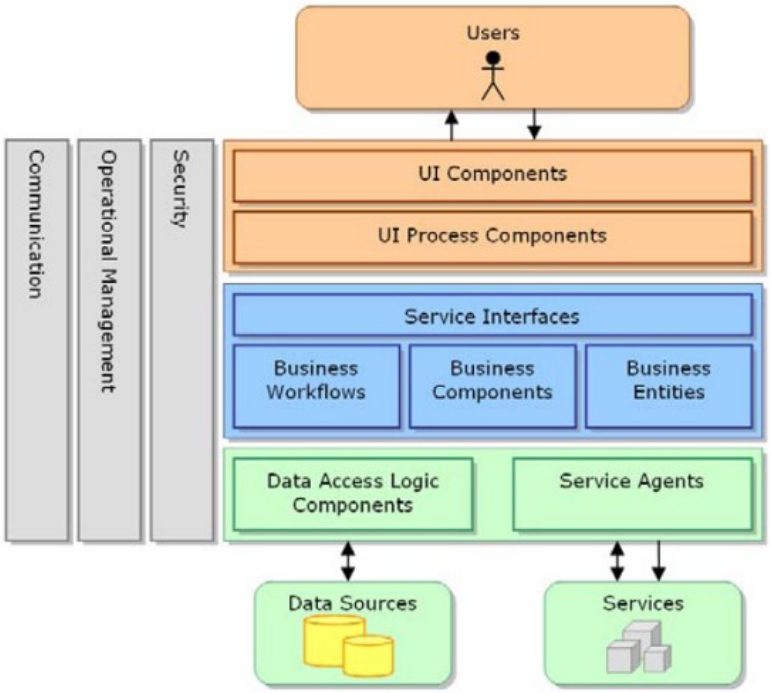


Figura 7: Arquitetura Lógica do cmNavigo

quatro camadas distintas. O acesso às bases de dados e aos serviços que alimentam o sistema é feito através componentes específicos, formando uma das camadas do sistema. Uma outra camada é a interface do utilizador, que engloba todos os processos e componentes que são visualizados por este. A terceira camada é a camada intermédia onde é efetuado o processamento dos dados da camada inferior para serem visualizados pelo utilizador na camada superior. Por fim, existe uma camada que é transversal a todas as anteriores que cuida de toda a segurança, comunicação e gestão operacional.

A arquitetura física do cmNavigo pode ser visualizada na Figura 8. Todo ele é constituído por tecnologias Microsoft, desde a interface do utilizador desenhada em C# e Silverlight, passando pelos objetos de negócio, que são todos os objetos base mais os objetos possíveis de criar pelo utilizador para simular a sua linha de produção, concebidos em C# e .Net e terminando nas bases de dados desenhadas em SQL.

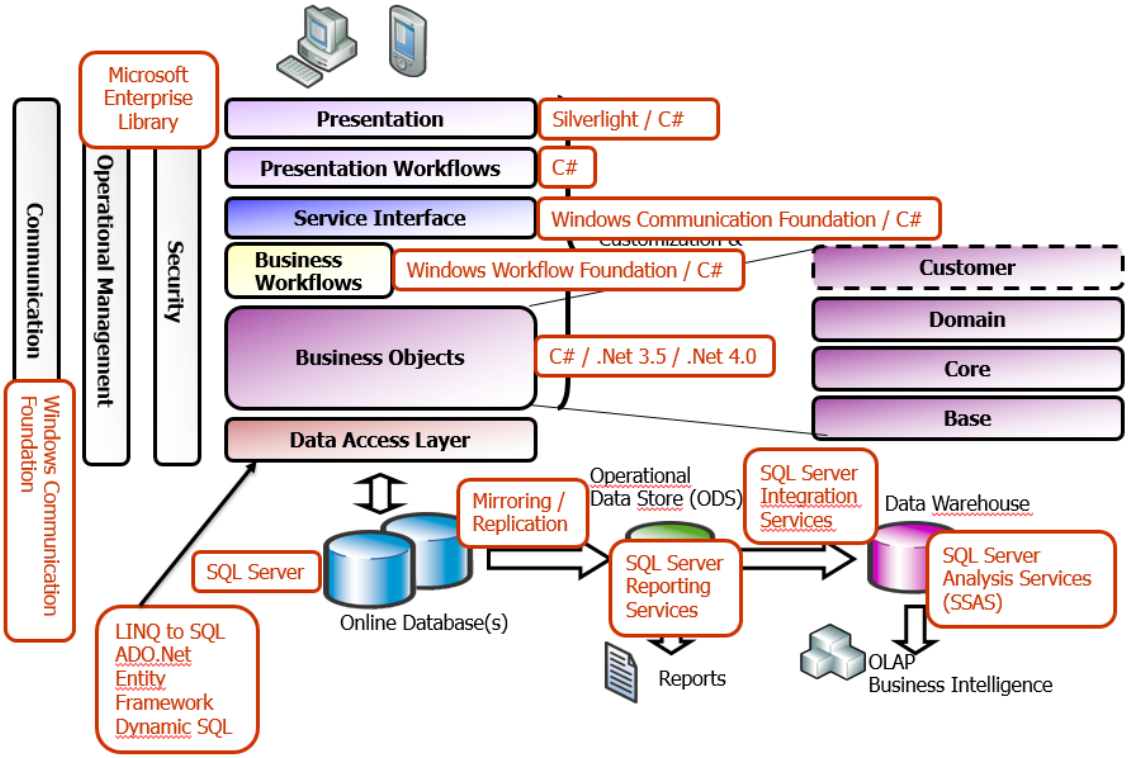


Figura 8: Arquitetura Física do cmNavigo

2.3.2 Módulos

Os módulos são, como já foi referido, conjuntos de funcionalidades. O cmNavigo apresenta mais de quarenta módulos que estão divididos em cinco categorias: Visibilidade em Tempo Real, Eficiência Operacional, Qualidade Integrada, Business Intelligence e Integração Fabril. De seguida são apresentados os principais módulos de cada categoria. [MCP14]

2.3.2.1 Visibilidade em Tempo Real

Aqui estão agrupados os módulos que permitem monitorizar todos os recursos e visualizar graficamente e em tempo real a linha de produção. [MCR14]

- **Fablive**, que possibilita uma visão imediata sobre o estado das linhas de produção assim como permite identificar rapidamente as áreas problemáticas;
- **Gráficos**, que podem ser facilmente criados por utilizadores com permissões para efetuar *queries* sobre materiais ou recursos, com informações em tempo real;
- **Relatórios** variados, incluindo OEE, UPH, Cycle-time, Yield e Uptime, formando um conjunto inovador e completo de relatórios de Engenharia Industrial;
- **Genealogia do Material**, acedendo facilmente às informações do histórico, completo ou filtrado, de qualquer objeto.

2.3.2.2 Eficiência Operacional

Neste módulo estão todas as ferramentas necessárias à gestão avançada de todos os recursos existentes na linha de produção. [MCE14]

- **Materiais e Contentores**, que inclui a lista de materiais com o consumo manual ou automático de matérias-primas e peças;
- **Coleção de Dados**, com limites flexíveis, amostras dinâmicas, grupos e cálculos associados;
- **Rastreabilidade de Equipamento**, onde é possível monitorizar todo o *cluster* de equipamentos e vários modelos de estado definido pelo utilizador;
- **Gestão da Manutenção**, definida por tempo e/ou utilização de equipamento que, por estar totalmente integrada, permitem a definição rigorosa de ações de manutenção.

2.3.2.3 Qualidade Integrada

Inclui os módulos que permitem a gestão do ciclo de vida do produto, assim como o controlo estatístico dos processos e a melhoria contínua das operações. [MCQ14]

- **Catálogo Eletrónico de Falhas**, que liga tipos de falhas às imagens e é onde o processo de definição obtido através do catálogo de falhas é gerado;
- **Gestão de Especificação**, que é onde é modulada toda a fábrica e são geridos todos os dados mestre;
- **Gestão de Exceções**, onde o utilizador define os protocolos de exceção, programando-os para estes serem despoletados manual ou automaticamente.
- ***Checklists***, que podem ser sequenciais ou móveis, obrigatórias ou facultativas e com a opção de se incluirem parâmetros e se definir o nível de monitorização para cada passo.

2.3.2.4 *Business Intelligence*

Na *Business Intelligence* são reunidas informações obtidas a partir de análises de dados de engenharia obtidos através da extrações de dados de diversas fontes, consultas e módulos. [MCO14]

- ***Operational Data Store***, onde é feita a integração de fontes de dados externas e heterogêneas com base num motor de extração, transformação e carregamento;
- ***Data Analysis***, que é uma fonte heterógena de dados múltiplos e uma plataforma de automação;
- ***OLAP***, onde este tipo de operações, recorrendo a um armazém de dados, são feitas no sistema de forma interativa;
- ***Data Mining***, que é composto por análises e algoritmos avançados, incluindo séries temporais, árvores de decisão e redes neuronais.

2.3.2.5 Integração Fabril

Aqui é efetuada uma integração da plataforma de produtividade e do equipamento de linha de produção, e é ainda aqui a implementação das soluções ERP. [MCI14]

- **Integração com ERP**, onde são trocadas as ordens de produção, as atualizações de status de inventário, os dados mestre e os dados da operação de manutenção;
- ***cmConnect***, que é uma plataforma de integração de equipamentos e que permite total integração com o cmNavigo ou outro MES;
- ***cmFoundation***, uma outra plataforma de integração e produtividade desenvolvida na empresa que tem como objetivo integrar todas as camadas necessárias do sistema e aumentar a comunicação das aplicações de diversas indústrias de produção.

2.4 Sumário

Com este estudo inicial dos conceitos necessários a um melhor entendimento do problema foi possível perceber que tipo de trabalho relacionado é que será necessário pesquisar para saber como irá ser concebido o simulador para o cmNavigo.

A simulação é sem dúvida uma das melhores formas das empresas conseguirem ter ideia do impacto que uma simples alteração irá causar. Existem diferentes modelos de simulação que indicam a forma como a simulação irá ser feita, mas é nos paradigmas que está o verdadeiro problema, pois será a escolha de um bom paradigma ou do conjunto de dois ou mais, que ditará se o simulador será eficiente ou não.

Mais à frente será explicado de uma forma mais prática as funções do cmNavigo, tendo esta parte um intuito mais introdutório focando-se apenas numa explicação macro da ferramenta.

Capítulo 3

Revisão Bibliográfica: Trabalho Relacionado

Tendo feito um levantamento dos conceitos usados, serão agora apresentados os principais trabalhos relacionados sobre o tema. É possível dividir estes trabalhos em dois grandes grupos: trabalhos sobre a modelação de Manufacturing Execution System e trabalhos sobre a implementação de um simulador de MES, sendo que em alguns casos os projetos abordam as duas temáticas. Antes de desenvolver um simulador é necessário conceber um modelo de simulação para ele, e é nessa parte que os primeiros trabalhos se focam.

3.1 Modelação de MES

Existem várias abordagens para a modelação de um Sistema para a Execução da Manufatura: abordagens simples, onde é usado apenas um paradigma, sendo os casos mais utilizados o da Simulação Baseada em Agentes, e existem abordagens mais complexas onde são combinados dois ou mais paradigmas para diferentes partes do modelo. De seguida serão apresentadas os métodos que obtiveram melhores resultados no seu contexto.

3.1.1 Agent-based modeling and simulation of an autonomic MES

Uma das abordagens mais inteligentes relacionadas com agentes foi a de Rolón e Martinez. Usando a metodologia Prometeus combinada com a abordagem Hermes de criação de agentes inteligentes, foi possível criar um novo método muito mais robusto para a modulação do sistema, que é apresentado na Figura 9. [RMM12] [RME12]

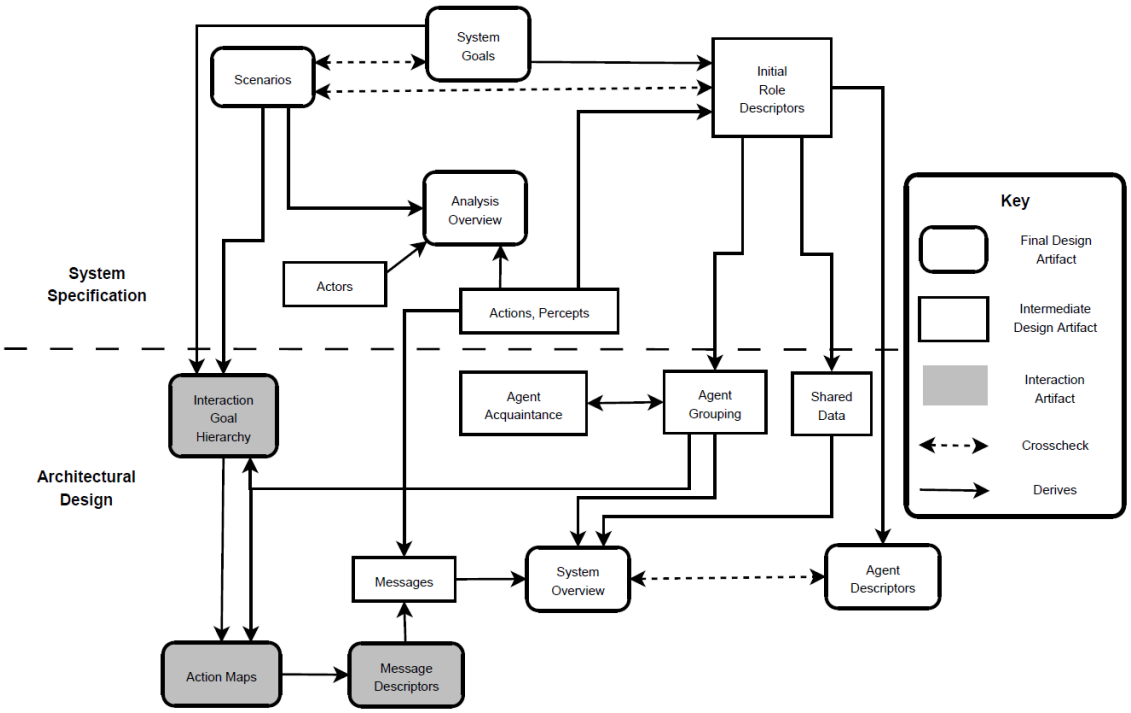


Figura 9: Metodologia Prometeus combinada com a abordagem Hermes

Prometeus é então uma metodologia de desenho detalhado que serve para especificar, conceber e implementar sistemas de agentes inteligentes. [PLM04] É diferente das outras metodologias do género porque permite o desenho de agentes inteligentes, suportando funcionalidades do início ao fim da execução do agente. O desenho do sistema é dividido em três partes como é possível ver na Figura 9 [CHD08] [RMM12]:

- **A especificação do sistema** que integra atividades como a identificação dos objetivos principais, a conceção dos casos de uso e a identificação das funcionalidades a serem implementadas;

- **O desenho da arquitetura** que consiste no agrupamento de algumas funcionalidades para determinar tipos de agentes relevantes, descrever a estrutura geral do sistema e desenvolver protocolos de interação dos casos de uso;

- E **o desenho detalhado** que inclui o desenvolvimento de diagramas de processos e de agentes (mostrando o funcionamento interno destes), a introdução dos planos para lidar com eventos e a definição detalhada desses mesmos eventos, dos planos e das crenças de cada agente.

Hermes é uma metodologia que contém uma abordagem sistemática para a criação de interações dirigidas para objetivos, afastando-se de metodologias centradas em mensagens como é o caso da Prometeus. Pode também ser separada em três passos principais: [CMW05]

• **Identificação das *Interaction Goals*** (IGs) e definição da sua **Hierarquia**, que são objetivos de alto nível que precisam de ser atingidos com sucesso;

• **Identificação das Ações** e **Sequenciação** das mesmas, onde cada ação é um passo que é preciso percorrer por um agente para atingir uma IG;

• **Identificação das Mensagens** e sua **Definição**, que são as mensagens trocadas entre agentes.

A principal diferença entre estas duas metodologias é que a primeira é focada na troca de mensagens enquanto a segunda é orientada para o cumprimento de objetivos, apesar de na metodologia Prometeus existir a definição de objetivos e na Hermes existir a identificação das mensagens a serem trocadas. É então focalizando mais os objetivos do sistema que esta combinação pode ser feita. Ao contrário de numa metodologia Prometeus pura, os objetivos do sistema estão diretamente ligados à hierarquia de IG, como pode ser visualizado na Figura 9.

Tendo então a metodologia certa, é possível passar à definição dos agentes. Neste caso foram agrupados dois grandes conjuntos de funcionalidades que formaram dois agentes distintos: os agentes das ordens e o agente dos recursos. [RMM12] [RME12]

O agente dos recursos possui o conhecimento que determina as tarefas relacionadas com o

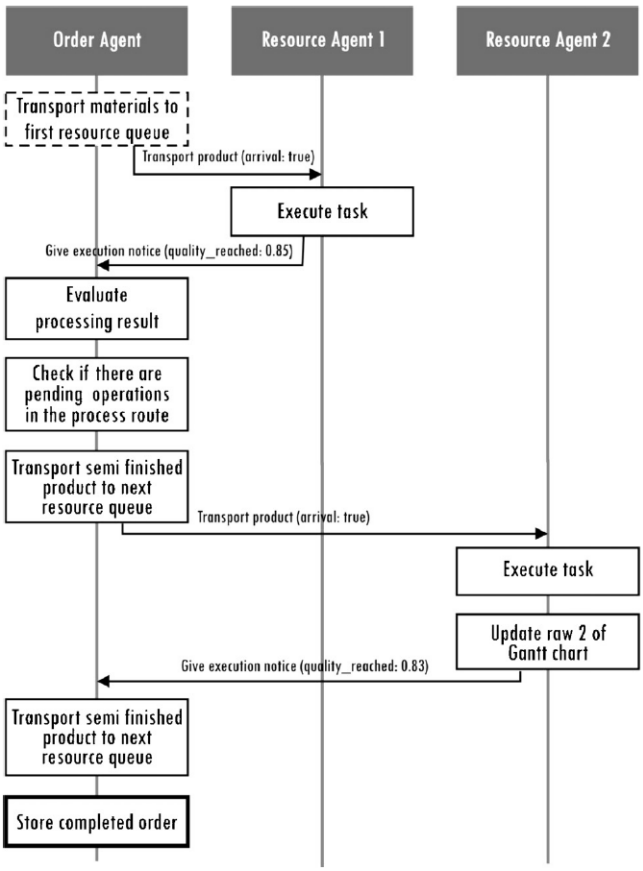


Figura 10: Modelo utilizando dois tipos de agentes

escalonamento e o controlo da execução. É ele que seleciona a próxima ação que o agente dos recursos irá efetuar. As principais funções deste agente são então verificar todas as operações que estão pendentes, gerar uma lista de possíveis soluções, escolher a melhor solução e indicá-la ao agente dos recursos. [RMM12]

Por sua vez, o agente dos recursos gera o escalonamento detalhado de um recurso, registando o seu correto ou incorreto uso num diagrama de Gantt. É também responsável pela execução das tarefas sobre um determinado recurso. Este agente tem como principais funções verificar a disponibilidade dos recursos, monitorizar e fazer update do planeamento e registar e executar as tarefas. É possível ver a ordem de trabalhos destes dois agentes na Figura 10. [RMM12]

Apesar do modelo criado ser bastante promissor, a criação de apenas dois agentes diferentes limita bastante as operações mais complexas que sejam necessárias efetuar, já que a CM tem parcerias com indústrias de alta tecnologia. Um bom ponto de partida para a utilização deste modelo seria a criação de um maior número de agentes para abringir todas as especificidades das linhas de produção com que o cmNavigo trabalha. [RMM12] [RME12]

3.1.2 Integration of Manufacturing Execution System and Simulation

Existem diversas vantagens e desvantagens em usar cada um dos paradigmas ou dos modelos de simulação. Seria então ideal tentar formar um modelo que aproveitasse as vantagens de mais que um método de maneira a formar um novo método mais eficiente. Foi essa a ideia de Worapradya e Buranathiti ao juntarem Eventos Discretos com Simulação Contínua. [WKT00]

Tendo por base o caso de estudo de uma indústria de produção de aço, o primeiro passo foi então a identificação da maneira ideal de conceber o modelo de simulação. Tentando aproveitar os benefícios do uso de múltiplos paradigmas, dividiu-se o modelo em duas fases:

- **Fase das Operações**, onde é utilizada a modelação por eventos discretos para se conseguir representar todas as ordens de despacho dos produtos, assim como toda a informação detalhada da produção. É assim possível otimizar o planeamento geral da produção que inclui a o plano das operações e a capacidade de produção. [WKT00]

- **Fase de Desenvolvimento**, onde se usa a simulação contínua para modelar o desenvolvimento e a modificação do projeto, incluindo a conceção, comissionamento e manutenção do MES. É nesta fase que a otimização e a execução da produção é simulada, assim como é ainda feita a monitorização dos materiais e a coleção de todos. [WKT00]

Esta divisão é feita porque os eventos que ocorrem na primeira fase são separados por horas, dias ou até semanas (o plano de produção pode mudar todos os dias ou de semana a semana, por exemplo), enquanto na segunda fase a precisão dos cálculos nos processos das

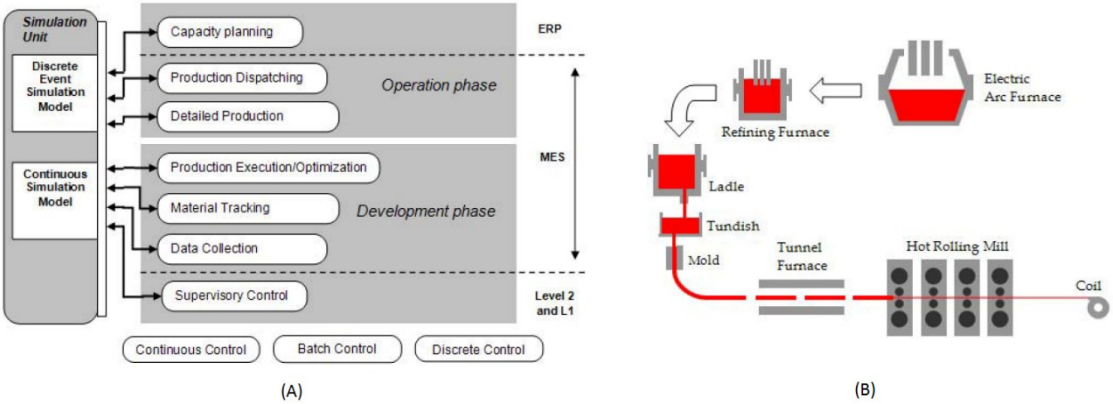


Figura 11: (A) Separação da modelação e (B) Linha de produção de aço

máquinas chega até aos milissegundos. É possível ver na Figura 11 em detalhe como a divisão é efetuada. [WKT00]

Na Figura 12 é possível ver então o modelo final já adaptado ao caso de estudo. Para perceber este esquema, é importante ter uma pequena noção de como é o processo de fabricação do aço. Existem duas fases importantes: a moldagem do aço e o arrefecimento do aço. Como este é um processo repetitivo, existe ainda uma parte do módulo que alterna entre estas duas fases, chamada de controlador. Como se está a falar de um modelo de simulação matemático, é possível definir equações que serão utilizadas na simulação. Por exemplo a equação do modelo da moldagem do aço:

$$\frac{dx_m(t)}{dt} = \frac{(ax_v + b)c\sqrt{2gh} - A_m v_c(t)}{A_m}$$

onde $x_m(t)$ é o nível da moldagem do aço em relação ao tempo, A_m é área do corte transversal do molde, v_c é a velocidade do casting (fundição) do aço, a e b conseguem-se determinar medindo a área de entrada de duas válvulas diferentes, c é o coeficiente de descarga, h é a altura mais ou menos constante de matéria no repartidor e x_v é o fluxo de aço que entra no sistema. [WKT00]

Esta é uma abordagem que é muito direcionada apenas para uma indústria. Se o objetivo fosse simular um Manufacturing Execution System que apenas trabalhasse para uma indústria, talvez fosse uma boa abordagem de maneira a personalizar o problema. Como o MES em questão tem de conseguir funcionar com várias indústrias, que podem ter linhas de produção muito diferentes, faz sentido criar um simulador que seja mais descentralizado do processo da linha de produção. [WKT00]

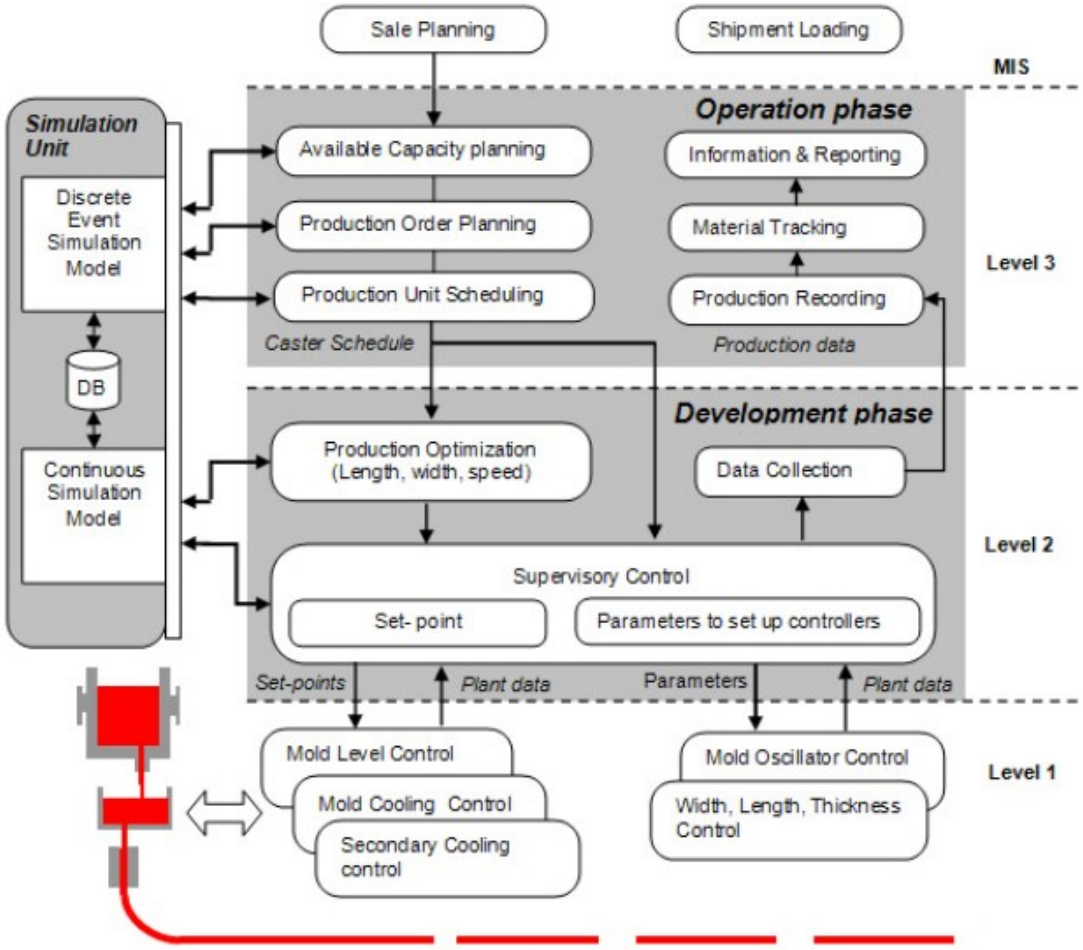


Figura 12: Modelo da Linha de Produção de Aço

3.1.3 Modeling and Simulation Approach for an Industrial MES

Um dos raros casos onde a modelação e a simulação são feitas no mesmo projeto é o trabalho de 2013 de Rabbani, Khan, Ahmad, Baladi e Naqvi. Usando como caso de estudo uma linha de produção bastante simples de pintura industrial e recorrendo à simulação por eventos discretos, conseguiram conceber o diagrama de estados apresentado na Figura 13. [RMJ13]

No primeiro estado – Start Upp – o MES vai incializar certos estados do sistema de acordo com os dados modelados que lhe são chegados. Check Errors verifica se não existe nenhum

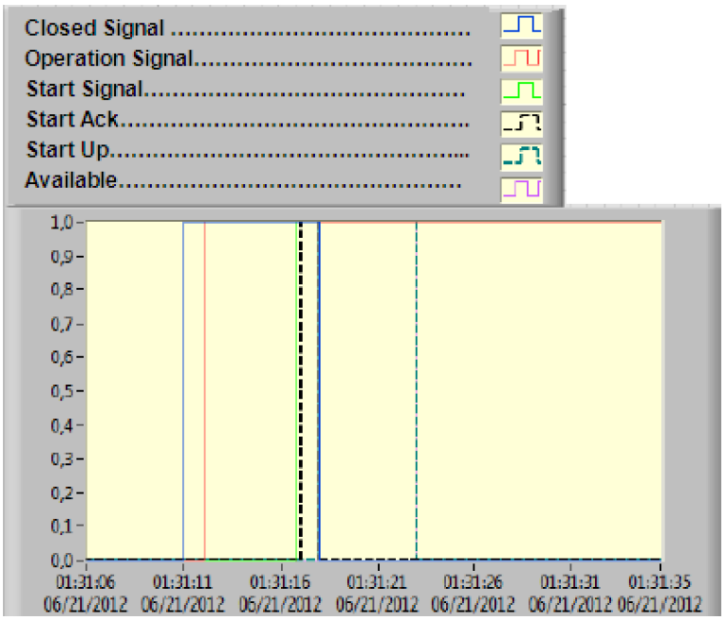


Figura 13: Resultados da simulação efetuada

alarme (um dos dados que é passado ao estado Start Upp) que impossibilite a continuação do processo. Se existir, é chamado o estado Fail, senão o processo passa para o Start Cmd. O programa principal é então iniciado quando o botão Start é acionado. [RMJ13]

A transmissão de informações entre o MES e o simulador é feita por sinais, que não passam de dados que foram modelados. Os sinais que são enviados do MES são sinais de ação e os sinais que são enviados do simulador são sinais de reconhecimento.

Neste caso a simulação trata apenas de saber como é que a troca de sinais ocorreu entre o MES e a unidade de simulação, não registando mais qualquer tipo de informação, tal como é mostrado na Figura 14. Não apresenta muito detalhe de como o simulador foi construído, por isso a informação importante a retirar deste trabalho é que é possível construir um modelo de simulação para um MES e de seguida conceber o simulador para este. [RMJ13]

É ainda importante referir estudos não sobre a escolha do modelo mas sim sobre o desenho do mesmo. Neumann, Westkämper e Constantinescu, em dois estudos diferentes, abordam dois métodos de modelar e simular sistemas de montagem: Multiscale Modeling e Situation-Based Modeling. Enquanto o primeiro é direcionado para a resolução de problemas físicos que apresentam características importantes em múltiplas escalas, particularmente espaciais e/ou temporais, o segundo é uma abordagem para simulações baseadas em eventos de alto nível – as situações. Algo comum nos dois estudos são quatro dos cinco pilares inicias para o desenvolvimento do modelo, que é a informação importante a retirar daqui: [NEM12] [NMW13]

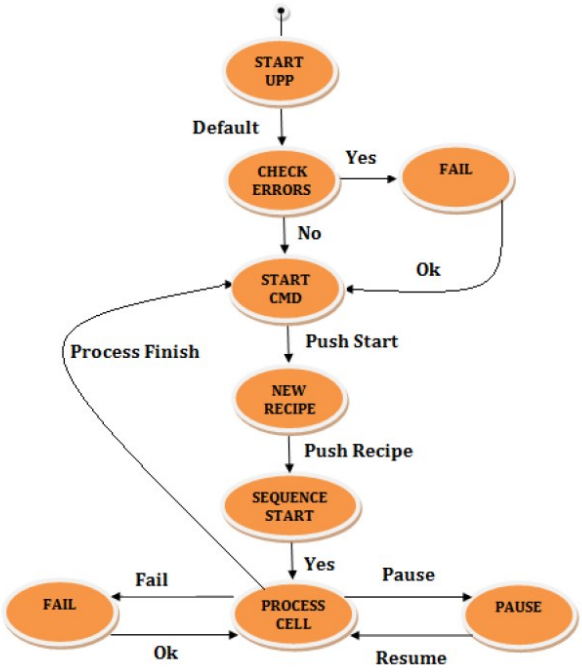


Figura 14: Diagrama de Estados do MES

- **Modelo base do sistema de montagem**, que contém os objetos da fábrica e as interdependências entre eles; [NEM12] [NMW13]
- **Bilioteca de recursos**, que consiste em objetos da fábrica específicos para cada escala do sistema de montagem; [NEM12] [NMW13]
- Uso de **linguagens de modelação** que são próprias para o efeito. Consistem de elementos e instruções de modelação para descrever um sistema e apoiar a estrutura, servindo para representar e visualizar uma parte específica de um sistema; [NEM12] [NMW13]
- Definição dos **procedimentos de modelação**, que são as sequências que permitem, passo a passo, a geração do modelo. Assim as atividades planeadas e a informação e objetos necessários para cada passo do modelo são tratadas efeicazmente. [NEM12] [NMW13]

3.2 Simulação de MES

Após se ter definido o modelo, o passo seguinte é então a simulação. A simulação não é nada mais do que computar o modelo que foi definido. De seguida serão apresentados os estudos sobre a criação de simuladores que mais se relacionam com o problema.

Uma maneira de construir um bom *software* é ter em atenção os trabalhos no mercado relacionados, por isso é também feita uma análise dos principais simuladores industriais.

3.2.1 Image-Scenarization: From Conceptual Models to Executable Simulation

Um dos melhores estudos sobre o tema é o de Rioux e Lizotte que mostra uma abordagem para a criação do processo iterativo de geração de agentes executáveis. É também discutido como é que o modelo baseado em agentes é criado usando vocabulário de *Scenarization*, que é constituído por alguns conceitos [LMF10] [RFM11]:

- **Agente**, um ator que tem regras de motivação para cumprir ações e modificar o sistema e cujas reações, que cumprem regras reflexas, também o modificam;
- **Paciente**, outro ator que não tem ações mas apenas reações, que também estão sobre a alçada de regras reflexas e modificam o sistema;
- ***Decor***, um objeto que faz parte do ambiente mas não é um ator mas que pode interagir com eles;
- **Ação**, um comportamento de um Agente;
- **Reação**, um comportamento de um ator, Agente ou Paciente;
- **Variável**, um atributo que é modificado pela simulação;
- **Parâmetro**, um atributo que nunca é modificado;
- **Predicado**, uma regra que é composta por uma entidade, uma relação e por um atributo ou um comportamento;
- **Ciclo**, um construtor que especifica a ordem por qual os comportamentos são executados;
- **Cenário**, um construtor que contém Parâmetros, todos os atores que fazem parte da simulação e a definição de todos os Ciclos.

Esta é apenas mais uma forma de representar um modelo de simulação tendo por base a modelação baseada em agentes, a parte realmente inovadora neste projeto é a aplicação de uma *framework* que permite aos agentes interagirem entre eles como consequência dos seus comportamentos, de seu nome GABS. Os objetos desta framework em tudo se assemelham aos objetos do modelo acima apresentado, sendo que a principal diferença é que cada objeto tem mais informação. O metamodelo do cenário é expresso num esquema XML e é compilado pela biblioteca JAXB. Na Figura 15 é possível ver como tudo se monta. [LMF10] [RFM11]

CoGUI é uma ferramenta de construção de grafos conceptuais. O modelo é desenvolvido nesta ferramenta e, quando o gerador do cenário recebe o modelo concetual intermédio, consegue extrair daí entidades, comportamentos, ciclos e componentes e criar as classes Java correspondentes. O código fonte é gerado através da ferramenta Apache Velocity, possibilitando que sejam então adicionadas funcionalidades, através do uso do Eclipse por exemplo, e tendo no

fim as classes que serão usadas no GABS para constituir os objetos. Ao mesmo tempo são criados os objetos XML nativos da *framework* que irão conter o estado inicial da simulação. [LMF10] [RFM11]

Neste projeto são introduzidos alguns fatores que ainda não tinham sido relatados em mais nenhum estudo. O uso de uma ferramenta externa para a criação do modelo conceptual visualmente que permite gerar código fonte Java é uma boa forma de se salvar tempo, apesar do código que é gerado não estar otimizado. Outro fator é a separação do modelo de dados da implementação em si, uma boa prática de *software* que pode ser seguida. Apesar de tudo, a ideia da *framework* não pode ser adaptada pois, tal como os autores referem nas conclusões: *"está longe de estar perfeita"*. Outro dos motivos é que o objetivo não é que o utilizador crie um modelo concetual antes de acontecer a simulação, e sim que forneça apenas alguns dados. [LMF10] [RFM11]

Existem ainda alguns trabalhos que valem a pena ser mencionados, como o de Iassinovski,

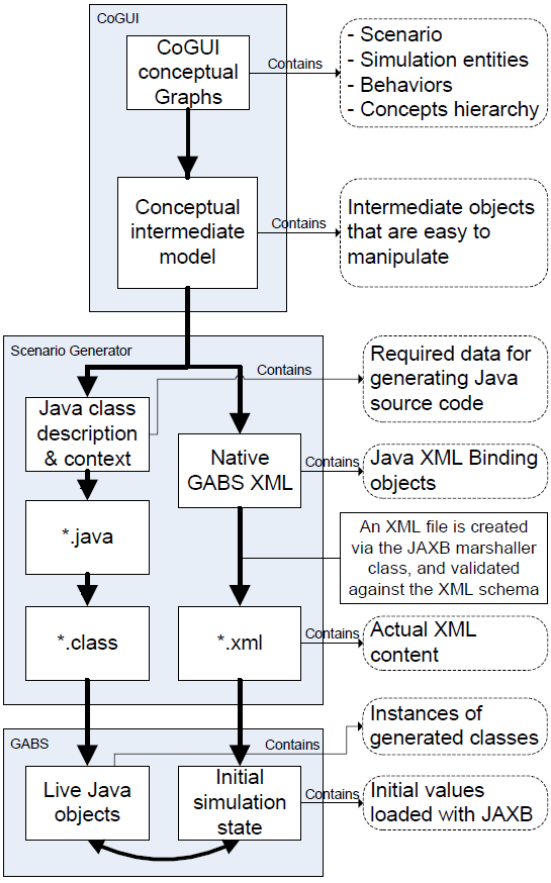


Figura 15: Da concepção à simulação

Artiba e Fagnart onde é criado um protótipo visual de um sistema baseado em regras para a simulação *online* e a aplicação WebManSim, um protótipo baseado em tecnologias web para a simulação de sistemas de manufatura. Apesar do primeiro não estar diretamente relacionado

com a simulação de MES, é possível perceber como é que um simulador visual é concebido. Já o segundo estudo é sobre uma abordagem leve para a criação de um simulador *online* e ajuda a entender os passos que têm que ser dados para criar um simulador leve, requisito fundamental deste trabalho. [RAY08] [KEH09]

3.2.2 Simuladores Industriais

De forma a entender quais os principais requisitos, foi feito um estudo dos principais simuladores industriais presentes no mercado. Dois deles destacaram-se por serem líderes; o Arena e o Simul8. A principal semelhança entre estes dois *softwares* é que ambos usam a modelação baseada em eventos discretos. Até agora foi assumido que teria que ser construído um simulador de raiz, mas e se algum destes *softwares* já disponíveis no mercado pudesse simplesmente ser adaptado ao cmNavigo? O projeto mudaria de rumo passando a ser apenas a adaptação de um simulador industrial a um Manufacturing Execution System já existente.

Este passo é impossível de ser feito porque o objetivo final é integrar com o próprio cmNavigo o simulador a ser realizado. Caso se usasse um simulador externo, além de implicar mais custos ao cliente porque iria precisar de duas ferramentas e não apenas de uma, nunca se obteria a mesma relação de interação, a mesma "cumplicidade" que se pode obter quando um simulador é feito especificamente para essa ferramenta.

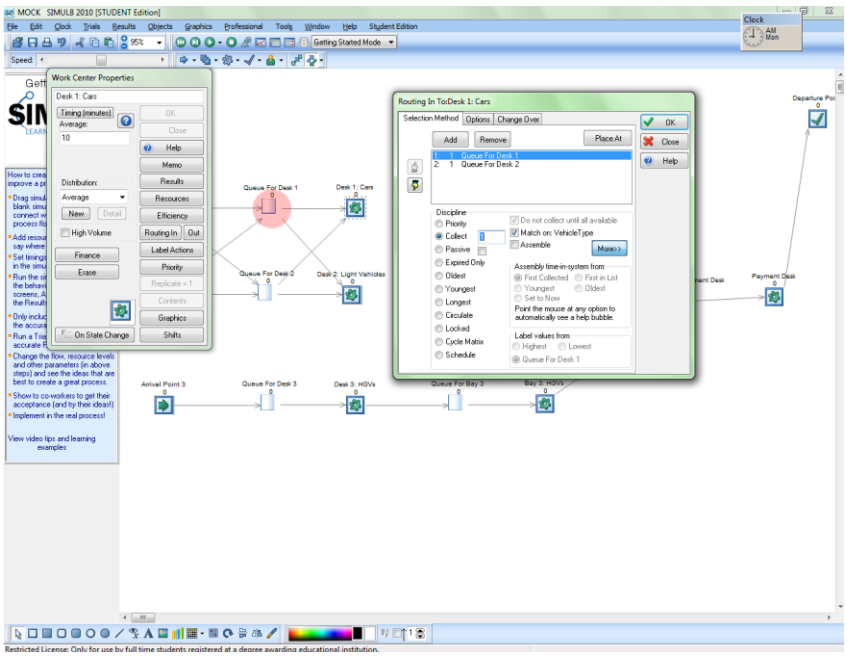


Figura 16: Ambiente de desenvolvimento do Simul8

Apesar de tudo é essencial um breve estudo das ferramentas existentes para se perceber quais são as melhores técnicas a adotar na construção gráfica da simulação. Estudo esse que será sobre o Simul8 e será aprofundado aquando do desenvolvimento do simulador.

3.2.2.1 Simul8

O Simul8 é uma ferramenta para planear, conceber e otimizar a produção de um produto, uma manufatura ou a logística de um serviço. Permite ao utilizador a criação de um modelo que é semelhante ao equivalente real do que está em estudo. Este modelo não é programado ou criado através de dados, mas sim através do desenho de esquemas representativos. Na Figura 16 é apresentada uma execução do Simul8. [COS14]

Tem como principais componentes:

- **Work Itens**, que são elementos ou entidades que entram no sistema e induzem diferentes atividades, usam diferentes recursos e no fim deixam o sistema. São exemplos disso um produto, um documento ou um cliente.
- **Entradas**, que são objetos que representam a entrada de entidades no sistema, como por exemplo a chegada de um cliente;
- **Atividade**, objeto que modela ações pelas quais uma entidade passa. É aqui que os recursos são usualmente modificados;
- **Fila**, objeto que simula a acumulação de entidades num determinado ponto da execução, normalmente por falta de recursos;
- **Saída**, que é o ponto pela qual as entidades deixam o sistema;
- **Recurso**, objeto que é usado para as restrições de capacidade de modelagem de trabalhadores, materiais ou meios de produção utilizados nas atividades;
- **Rotas**, que são objetos que ligam todos os outros objetos presentes no modelo de simulação e representam sequências de atividades.

O Simul8 é apenas um dos vários simuladores industriais disponíveis no mercado. No Anexo B encontra-se uma tabela com os principais simuladores atualmente presentes no mercado. [SMA10]

3.3 Sumário

O estudo dos trabalhos relacionados levantou um problema: a inexistência de um simulador para um Manufacturing Execution System no mercado e que possa ser usado por indústrias de alta tecnologia. Existem poucos estudos sobre o tema, e os que existem focam-se mais na criação do modelo de simulação do que propriamente na execução da simulação em si. Por isso foi necessário recorrer aos simuladores industriais existentes no mercado que mais uma vez levantaram um problema: criam visualmente o modelo de conceptual em vez de recorrerem a um conjunto de dados fornecidos pelo cliente. Esse será o principal obstáculo pois é algo que ainda não foi feito e é por isso que foi dedicada uma parte tão extensa da revisão do estado da arte aos conceitos.

Capítulo 4

Abordagem Metodológica

Neste capítulo será explicada em detalhe a abordagem utilizada neste projeto. É feita uma breve introdução, seguida da escolha da solução, incluindo a abordagem que foi utilizada para a realização do modelo de simulação, as tecnologias que serão utilizadas no decorrer do projeto e o trabalho que foi realizado até à implementação. Após isto é então explicado como será efetuada a validação da solução final e o planeamento das tarefas realizadas ao longo do decorrer da dissertação. No fim é apresentado um sumário do que foi discutido neste capítulo.

4.1 Introdução

Tal como já foi abordado, existiam duas formas de executar o projeto: encontrar um simulador que conseguisse cumprir todos os requisitos da CM e fosse possível adaptar ao cmNavigo ou então construir um de raiz. Uma das abordagens implicaria um corte significativo na dificuldade do projeto, por isso fez sentido começar a investigação pela pesquisa dos vários simuladores atualmente disponíveis no mercado, verificando se algum destes conseguia cumprir todos os requisitos propostos. Para validar essa decisão, foi criado um documento que pode ser consultado no Anexo F e que foi apresentado e discutido com a equipa do cmNavigo.

Existem sim muitos e variados simuladores industriais, tal como foi descrito no capítulo anterior, que são capazes de simular com muito detalhe uma linha de produção, mas nenhum deles consegue simular as operações de um MES sobre essa mesma linha. Por isso foi necessário abandonar o caminho mais fácil e seguir então pela criação de um simulador personalizado.

Segundo o estudo efetuado nos capítulos anteriores, a criação de um simulador pode-se dividir em dois passos: a criação do modelo e a execução deste. A criação do modelo pode ser

considerada tão ou mais importante que o desenvolvimento do código do simulador, pois se o modelo é corretamente criado, é robusto, terá um impacto positivo maior na performance da simulação que qualquer funcionalidade que possa ser implementada no simulador.

4.2 Escolha da Solução

O estudo inicial foi então de encontro à conceção do melhor modelo de simulação possível para o projeto. Numa primeira fase foram encontradas duas maneiras distintas de atacar o problema:

- Recorrendo apenas a agentes;
- Dividindo o problema em três paradigmas diferentes: agentes, eventos discretos e sistemas dinâmicos.

Após um estudo mais extensivo do cmNavigo que não tinha sido possível efetuar numa primeira fase, ficou clara a decisão: os eventos teriam que ser obrigatoriamente incluídos, já que este sistema já tem uma forma de distinguir claramente os passos que são efetuados numa linha de produção que se aproxima muito com a definição de um evento discreto: o *Step*. Aliado a isto, o conceito de agentes foi também introduzido pois é uma oportunidade de combinar os eventos usados pela maioria dos simuladores com os agentes que mostram bastantes progressos neste ramo. Após explicar um pouco melhor a abordagem, é também importante referir as tecnologias que serão usadas durante a realização do projeto e demonstrar o trabalho que foi efetuado antes da implementação do projeto que, além do estudo que já foi explicado e apresentado, passou pela elaboração de um Relatório de Requisitos e de um Relatório de Arquitetura, validados com a Critical, para melhor se perceber quais os objetivos concretos da simulação.

4.2.1 Abordagem Escolhida

Ao contrário dos outros paradigmas, um modelo baseado em agentes pode simular qualquer tipo de operações e processos, desde situações de grande abstração até dados específicos como objetos individuais com tamanhos distâncias ou velocidades. [CHY11] A única situação em que os agentes não se destacam é quando é necessário uma simulação contínua para efetuar com precisão o registo de dados ou sequência de operações na simulação. Para isso será necessário usar um paradigma de simulação diferente que consiga cobrir essa falha, neste caso concreto o modelo Baseado em Eventos Discretos. Mas como já foi visto, este modelo é muito centralizado na indústria, não deixando grande espaço para processos de mais alto nível. Por isso foi desenvolvida a hipótese da combinação de dois paradigmas que permitam entre si colmatar as falhas uns dos outros, semelhante ao modelo usado com algum sucesso pelo

AnyLogic, com a diferença de não ser incluído o modelo de Simulação Dinâmica. [ANY14] [FRA08]

Os dois paradigmas a usar são então:

• **Eventos Discretos**, o modelo atual mais usado na criação dos simuladores industriais pela sua facilidade de simulação. Estes eventos modelam a operação de um sistema como uma sequência discreta de acontecimentos no tempo. Cada evento ocorre num momento específico e marca uma mudança de estado no sistema. Assim, a maneira ideal de modelar processos ou sequências de operações é através de eventos. [BAF04]

• **Agentes**, sendo através deles que a descentralização do processo é feita. Tirando uma grande carga dos agentes com a introdução do paradigma anterior para modelar uma série de operações mais relacionadas com a logística, os agentes ficam livres para tratarem das operações fabris, não sendo necessário a criação de tantos agentes como nas hipóteses estudadas. Este tipo de modelação serve para simular as ações e interações dos agentes autônomos (tanto individuais como coletivos, como as organizações ou os grupos) com o objetivo de avaliar os seus efeitos sobre o sistema como um todo. [JRF12]

A abordagem seguida aproveita então dois objetos chave do cmNavigo: O *Step* e o *Material*. Enquanto um *Step* já é claramente um Evento dentro do cmNavigo, o *Material* foi adaptado para criar o conceito de agente. A implementação básica será a criação de uma *thread* para cada *Material*, onde este seguirá os *Steps* que lhe estão atribuídos, tomando depois decisões próprias e interagindo com outros agentes (*Materials*) quando necessário.

Quando a simulação é iniciada, os *Materials* que até agora estavam apenas no modelo são criados no cmNavigo e ficam disponíveis para serem simulados. Nesta altura, é lançada uma thread para cada *Material* que implementa um ciclo. Esse ciclo representa um *Step* e inclui todas as operações possíveis de realizar. A *thread* só irá terminar quando o *Material* terminar também, sendo que isso só acontece em duas ocasiões:

- Todos os ciclos (*Steps*) foram percorridos e o *Material* acabou a sua simulação;
- Algum erro ou falha forçou o *Material* a terminar prematuramente, terminando também o *thread*. Os erros vão desde, por exemplo, falta de *Resources* para completar a simulação até algum erro de comunicação com a ferramenta. Neste caso foi implementado um timeout que, se atingido, termina o *Material*.

Um exemplo simples de uma interação multi-agente dá-se no momento em que um *Material* atualiza diretamente no cmNavigo os *Resources* que não estão disponíveis naquele *Track In*. Nessa altura, este envia uma mensagem aos outros *Materials* para que estes saibam que a lista que está no servidor está desatualizada e que têm de esperar pela lista mais recente. No fim do *update*, é enviada uma nova mensagem a todos os agentes para que estes possam consultar a lista. É nesta altura que a vantagem de ter agentes é particularmente benéfica, pois para o resto das operações são os eventos que ocupam o lugar de destaque.

4.2.2 Tecnologias

Como já foi várias vezes referido, o objetivo final é a integração do simulador na ferramenta cmNavigo. Assim, o desenvolvimento do simulador terá que ser feito tendo em conta as tecnologias usadas na ferramenta de modo a ser possível efetuar a já falada integração. O cmNavigo foi desenvolvido com base em tecnologias Microsoft, por isso a solução final será desenvolvida usando essas mesmas tecnologias, nomeadamente C#, WPF, WCF e .Net, que podem ser consultadas em maior extensão e detalhe no Anexo E. Além destas tecnologias, uma prioridade será a forma como será visualizada a simulação. Já que o cmNavigo tem incluído um módulo de desenho e visualização chamado fabLive (Figura 17) será de valor usar essa mesma ferramenta para efetuar a visualização gráfica da simulação em tempo real, já que o fabLive o permite.

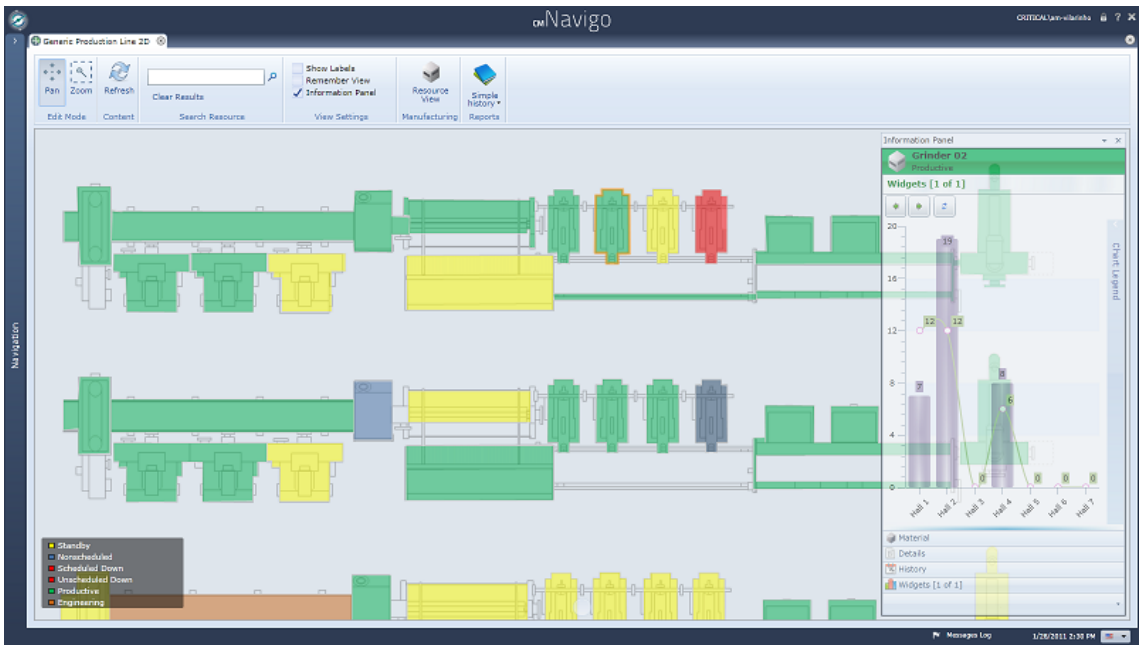


Figura 17: Screenshot da ferramenta fabLIVE

4.2.3 Trabalho Realizado

Além da pesquisa já apresentada e do documento sobre os simuladores, o trabalho que foi realizado pré-implementação passou pelo levantamento de requisitos junto da Critical e pelo desenho da arquitetura da aplicação final, elaborando dois documentos para o efeito: Relatório de Requisitos e Relatório de Arquitetura. Quando estes dois documentos foram validados, pôde então partir-se para o desenvolvimento da aplicação.

4.2.3.1 Requisitos

De maneira a ter uma melhor perceção das principais características do projeto, os requisitos foram catalogados segundo o método MoSCoW, que diz que um requisito deve ser obrigatório (*Must*), não é obrigatório mas deve ser cumprido (*Should*), não é obrigatório mas pode ser implementado caso haja tempo (*Could*) e finalmente pode ser feito numa futura *release*, mas não nesta (*Would*). Os requisitos da aplicação podem ser consultados em detalhe no Anexo D.

4.2.3.2 Arquitetura

Após a elicitação de requisitos do sistema é essencial organizar e projetar a arquitetura do sistema. Uma arquitetura bem definida torna-se essencial e ajudará na fase de implementação. O relatório desenvolvido tem como objetivo proporcionar uma melhor compreensão de alguns aspectos técnicos do projeto, tais como a forma como o sistema está estruturado e dividido. Também irá ajudar a entender quais e como as tecnologias discutidas estão interligados, assim como as suas características, obtendo assim uma visão geral da arquitetura do sistema. A arquitetura do sistema pode ser consultada em detalhe no Anexo E.

4.3 Verificação e Validação

Dois passos essenciais em qualquer projeto são a verificação e a validação do mesmo. [KJP92] Enquanto a primeira é uma verificação se o modelo está bem construído e desenhado, a segunda é uma validação do modelo em si, se aquele modelo é o modelo certo para aquele caso específico.

Os passos a seguir para a verificação são então:

- Pedir a um outro programador que esteja dentro do tema para verificar o código;
- Fazer um diagrama onde se registe cada ação que o sistema pode fazer e testar cada um desses acontecimentos;
- Fazer com que se imprimam no fim do código uma série de estatísticas sobre os parâmetros de entrada;

• Imprimir os parâmetros de entrada para verificar se não foram mudados aquando da execução do código;

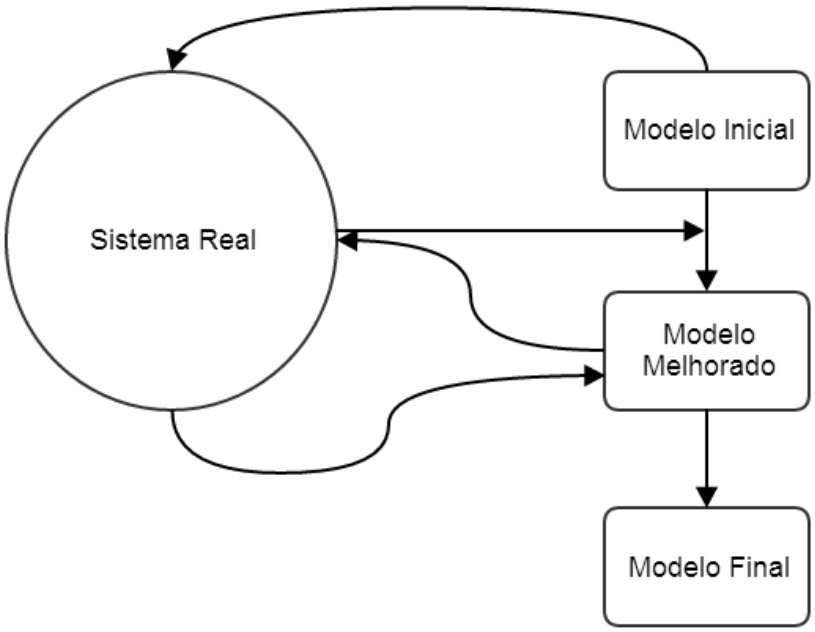


Figura 18: Validação de uma aplicação de simulação

• Tornar o código o mais auto-documentado possível para futuras verificações.

Em relação à validação, é necessário testar o código com vários parâmetros e várias vezes de modo a calibrar o modelo, como é possível ver na Figura 18. Este é um processo iterativo que só termina quando todas as condições finais estiverem cumpridas. Em qualquer projeto é necessária a validação do mesmo, mas neste caso esse passo é especialmente importante pois se na maioria dos projetos algum erro ou fraca performance não tenha um grande impacto no objetivo final, num simulador qualquer falha ou erro pode mudar completamente a simulação que se está a efetuar. Por isso serão efetuados dois passos diferentes na validação do simulador.

Numa primeira fase serão usados dados de teste fornecidos pela Critical. Esta primeira fase será efetuada num servidor próprio dentro da CM para teste do simulador. Exemplos dos dados a serem usados nesta fase podem ser encontrados no capítulo seguinte.

Na fase final o objetivo passa por utilizar dados reais de um cliente, de modo a refinar e otimizar o modelo de simulação. Esta fase é crucial pois um simulador só estará totalmente validado quando conseguir simular totalmente o ambiente de uma fábrica real depois de lhe serem introduzidos os dados que a caracterizam.

4.4 Sumário

A forma escolhida para executar a abordagem escolhida que já foi explicada em cima foi a adaptação da metodologia *Scrum*, onde existiram *sprints* de duas ou três semanas que no fim incluíam uma apresentação do trabalho que foi desenvolvido. É possível consultar o diagrama de Gantt dos *sprints* do projeto na Figura 30 do Anexo A. De seguida são apresentadas em detalhe as principais tarefas executadas ao longo deste para se conseguir obter uma melhor perceção do trabalho que foi efetuado.

4.4.1 Estudo do cmNavigo

- **Descrição**: Estudo da ferramenta desenvolvida na empresa de maneira a entender como será possível efetuar a integração do simulador.
- ***Inputs***: Documentação e acesso à ferramenta.
- ***Outputs***: Um melhor entendimento do cmNavigo.
- ***Sprints***: 1 (3 semanas)

4.4.2 Levantamento de Requisitos

- **Descrição**: Levantamento dos requisitos detalhados junto com a Critical para haver uma melhor compreensão das funcionalidades a implementar no simulador.
- ***Inputs***: Não tem.
- ***Outputs***: Relatório de Requisitos – Anexo D
- ***Sprints***: 1 (3 semanas)

4.4.3 Estudo das Tecnologias a utilizar

- **Descrição**: Após saber quais são os requisitos é necessário, a par com o desenho da arquitetura, escolher quais são as melhores tecnologias a utilizar na conceção da solução final.
- ***Inputs***: Conhecimento sobre o cmNavigo, requisitos levantados e arquitetura do sistema.
- ***Outputs***: Leque das tecnologias a utilizar e capítulo no relatório da arquitetura.
- ***Sprints***: 1 (3 semanas)

4.4.4 Desenho da Arquitetura

- **Descrição**: Conceção da arquitetura da aplicação, incluindo a arquitetura física e lógica, assim como elaboração de um relatório da mesma.
- ***Inputs***: Requisitos que foram acordados e tecnologias a utilizar.

- ***Outputs***: Relatório de Arquitetura – Anexo E
- ***Sprints***: 2 (2 semanas)

4.4.5 Desenvolvimento do Master Loader

- **Descrição**: Desenvolvimento de uma aplicação que seja capaz de criar ficheiros de MasterData utilizando os dados que extrai do cmNavigo.
- ***Inputs***: Aplicação já existente de carregar MasterData para o cmNavigo.
- ***Outputs***: Nova aplicação que faça também o inverso.
- ***Sprints***: 2/3 (3 semanas)

4.4.6 Desenvolvimento do Modelo

- **Descrição**: De maneira a se conseguir obter a melhor simulação possível, é necessário ter um modelo de simulação robusto por trás. Por isso é dedicado bastante tempo à escolha da melhor abordagem a seguir, assim como ao desenho do modelo segundo essa abordagem
- ***Inputs***: Abordagens a seguir e funcionalidades da aplicação.
- ***Outputs***: Modelo de simulação a ser implementado.
- ***Sprints***: 3/4 (3 semanas)

4.4.7 Desenvolvimento da Solução

- **Descrição**: Conceção e desenvolvimento do simulador e respetiva integração no cmNavigo.
- ***Inputs***: Conhecimento sobre o cmNavigo, requisitos e funcionalidades, arquitetura do sistema e modelo de simulação concebido.
- ***Outputs***: Simulador funcional.
- ***Sprints***: 4-8 (9 semanas)

4.4.8 Validação com Dados de Teste

- **Descrição**: É importante validar a aplicação com dados de teste para permitir perceber se cada módulo está a funcionar corretamente. Existirão duas fases de validação com dados de teste, sendo a primeira para a validação do modelo e a segunda para a validação do simulador.
- ***Inputs***: Dados de teste.
- ***Outputs***: Estatísticas sobre a performance da simulação.
- ***Sprints***: 4 (1 semana) e 8 (1 semana)

4.4.9 Validação com Dados Reais

- **Descrição**: Uma validação só estará completa quando for possível simular uma fábrica real com precisão. Assim serão efetuados testes à solução com dados de uma empresa real.
- ***Inputs***: Dados reais.
- ***Outputs***: Estatísticas sobre a performance da simulação.
- ***Sprints***: 8 (2 semanas)

Capítulo 5

Implementação

A implementação do protótipo é a parte central desta dissertação. Depois de todo o estudo efetuado, foram adquiridas bases suficientes para realizar com sucesso esta etapa. Neste capítulo é explicado passo a passo o que foi implementado, como e por que ordem. Inicialmente é explicada, com mais detalhe, a ferramenta para a qual o simulador foi implementado. A seguir são explicadas em detalhe as três aplicações diferentes implementadas: o Master Loader, o cmSimulatorConfig e o cmSimulator. No fim é apresentado um resumo do que foi discutido.

5.1 cmNavigo

Para conseguir explicar todo o processo da simulação implementado, é necessário primeiro perceber como funciona o cmNavigo de uma forma menos superficial. Já foi antes explicado o que era e quais as suas principais funções, mas ficou por explicar a sua execução propriamente dita.

De um modo mais prático, e sempre relativamente ao uso que o simulador lhe dá, o cmNavigo é uma ferramente de controlo de uma fábrica, que tem acesso tanto à informação dos planos de produção como das máquinas no chão da fábrica, conseguindo assim controlar se estes estão a funcionar em harmonia. Para configurar corretamente o cmNavigo é necessário fornecer então certas informações sobre a produção fabril usada. Estas informações servirão para popular a ferramenta, criando objetos e ligações entre estes. Estes objetos são denominados de *Business Objects* (ou Objetos de Negócio) e são todas as entidades presentes no sistema. Existem ainda outros tipos de objetos que fornecem informação adicional sobre os *Business Objects*, como é o caso dos *Entity Types* (que contém toda a meta-data dos objetos) até

diferentes tabelas (*Generic Tables*, *Lookup Tables* e *Smart Tables*) que contêm qualquer tipo de informação adicional.

Das dezenas de *Business Objects* que podem ser criados pelo cmNavigo, o principal é o *Material* já que toda a execução do programa gira à volta deste. De maneira a explicar melhor todo este processo, são apresentadas de seguida as principais entidades do cmNavigo. Na Figura 19 é possível ter uma ideia visual das ligações que estas têm entre si.

- ***Material***: O *Material* é um dos objetos mais importantes do sistema, uma vez que pode representar a matéria-prima, um *stock* ou um processo. Um *Material* também pode ser um

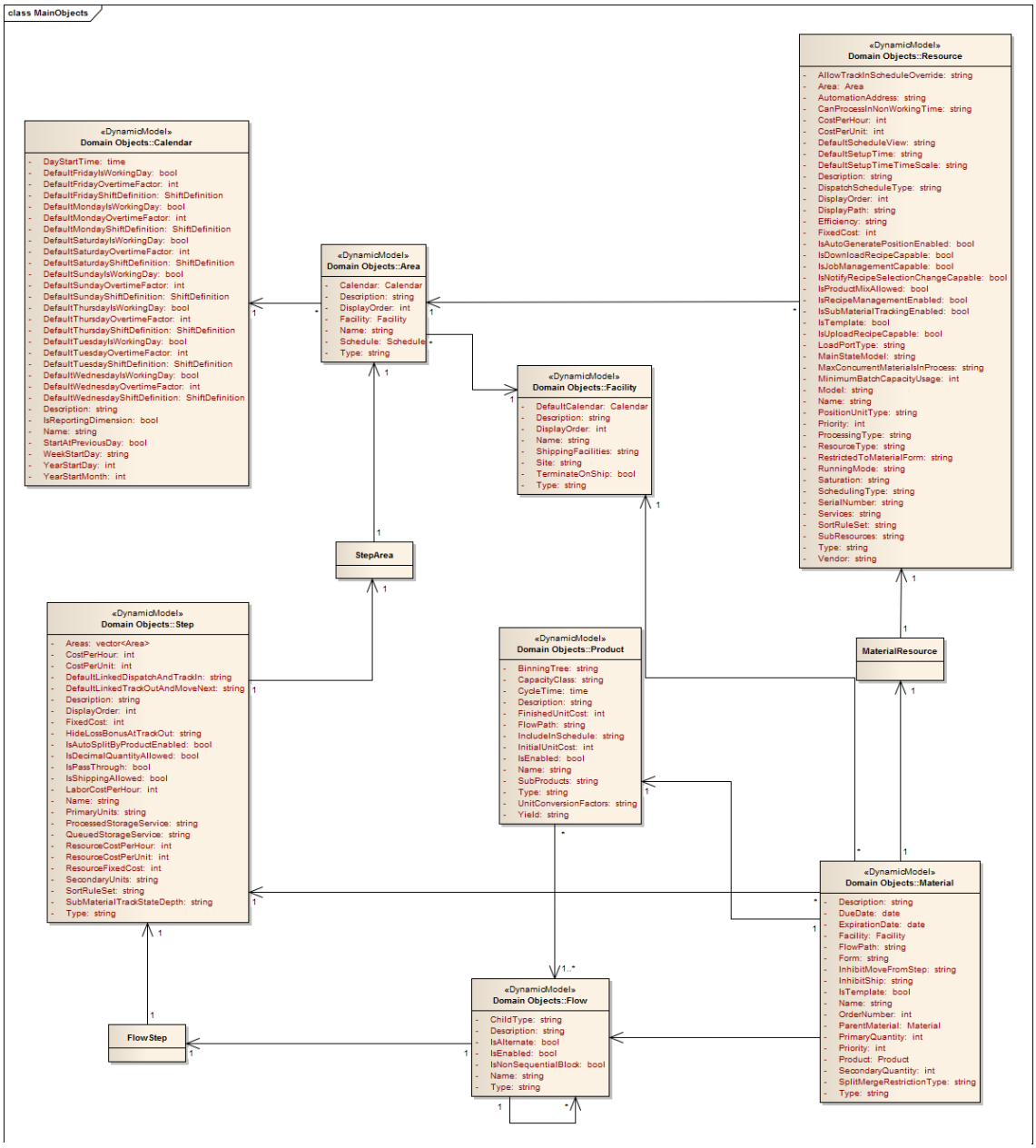


Figura 19: Modelo Relacional dos principais objetos do sistema

agregado de materiais. É a unidade mais básica da simulação, assim como é o seu centro. O objetivo principal é que a simulação corra 1 ou mais *Materials* pelos serviços do cmNavigo.

- ***Product***: Um *Product* é, tal como o nome indica, um produto final que é gerado naquela fábrica e determina as carecterísticas de um *Material* produzido. Durante todo o processo, um *Material* tem logicamente um *Product* associado.
- ***Resource***: Um *Resource* representa qualquer entidade que participa no processamento ou armazemento de um *Material*, seja equipamento ou pessoal. Existem 6 tipos de *Resource* diferentes: Process (processam um *Material*), Storage (armazenam e recuperam um *Material*), Consumable Feed (fornecem consumíveis a outros *Resources*), Load Port (serve como input, output ou input/output para outro *Resource*), Durable (um *Resource* que pode ser ligado a outro mas não pode ter nenhum material atribuído) e Component (pode ter vários *Resource* pais).
- ***Step***: Um *Step* representa uma operação do processo de fabricação, assim como a menor unidade do processo rastreável para o *Material*. Num *Step*, um *Service* é realizado num *Material*. Tendo o *Step*, o *Service* a ser executado pode depender opcionalmente do *Flow*, *Product* ou mesmo do *Material*. Passando para uma linguagem de simulação, um *Step* representa um Evento que acontece a um determinado *Material*.
- ***Flow***: Um *Flow* representa uma rota pré-definida ou o caminho para o *Material* e é uma combinação de outros *Flows* ou *Steps*. É importante destacar que um *Flow* só pode ter no mesmo nível *Flows* ou *Steps*, mas não ambos simultaneamente. Um *Flow* pode ser partilhado por vários *Materials* em qualquer momento, sendo que um *Material* tem sempre um *Flow* associado em qualquer ponto.
- ***Facility***: A *Facility* corresponde a um local físico, mais propriamente ao chão de uma fábrica, por isso os objetos físicos estão sempre associados a uma, como é o caso dos *Materials* e dos *Resources*.
- ***Area***: Uma *Area* é um agrupamento lógico de *Steps* e *Resources* que podem ter um local físico correspondente. Uma *Area* pertence sempre a uma *Facility*.

5.2 Master Loader

Já foi explicado para que serve esta informação e que tipo de informação é, mas como é então inserida no sistema? Existem duas formas: manualmente, na própria GUI do cmNavigo (tendo a desvantagem de ser um processo bastante moroso); ou carregada automaticamente através de um ficheiro excel próprio denominado de Master Data. Este excel tem toda a informação necessária para o sistema funcionar corretamente. Apesar de ser possível adicionar informação adicional mais tarde, a informação chave para o correto funcionamento do cmNavigo é possível de ser assim inserida. A grande vantagem da existência de um Master Data é a facilidade com que é possível modificar a informação e carregá-la posteriormente para o sistema, ficando assim acessível aos serviços do cmNavigo. Mas o objetivo da simulação é

modificar os objetos do sistema, ou seja, o Master Data que foi carregado inicialmente fica desatualizado e caso seja necessário repetir o processo noutra máquina, a simulação tem que ser feita de novo. A solução encontrada foi criar uma aplicação que efetuasse o processo contrário do que já existia: surgiu assim o Master Loader.

O Master Loader tem como principal objetivo recriar um ficheiro Master Data através da informação presente no sistema. Foi tirado proveito dos mesmos endpoints utilizados para carregar a informação para o cmNavigo, que utilizam o protocolo WCF e são o principal motivo por ser necessário implementar um simulador de raiz e não aproveitar um já existente. Faz sentido então explicar no que consiste um ficheiro de Master Data, assim como os objetos que fazem parte deste.

5.2.1 Master Data

Um ficheiro Master Data é constituído por 40 folhas de excel, sendo que cada folha corresponde a um tipo de entidade que é descarregada do cmNavigo. Cada uma dessas entidades pode ser de um tipo diferente: *Static Model*, *Dynamic Model*, *Generic Table*, *Smart Table* e *Lookup Table*. Além do tipo, cada entidade tem uma série de parametros próprios que formam o objeto, que não são nada mais do que a informação que se quer extrair do cmNavigo. No Anexo G é possível encontrar cada entidade descrita de uma forma mais detalhada. Na Figura 20 é possível ver um exemplo de uma folha de um ficheiro MasterData, neste caso a correspondente aos *Resources*.

	A	B	C	D	E	F	G
1	Name	Description	Type	IsTemplate	MainStateModel	Vendor	Model
2	StressResource001001	StressResource001001 Resource	Stress	No	SEMI E10 > Standby	StressVendor	StressModel
3	StressResource001002	StressResource001002 Resource	Stress	No	SEMI E10 > Standby	StressVendor	StressModel
4	StressResource001003	StressResource001003 Resource	Stress	No	SEMI E10 > Standby	StressVendor	StressModel
5	StressResource001004	StressResource001004 Resource	Stress	No	SEMI E10 > Standby	StressVendor	StressModel
6	StressResource001005	StressResource001005 Resource	Stress	No	SEMI E10 > Standby	StressVendor	StressModel
7	StressResource001006	StressResource001006 Resource	Stress	No	SEMI E10 > Standby	StressVendor	StressModel
8	StressResource001007	StressResource001007 Resource	Stress	No	SEMI E10 > Standby	StressVendor	StressModel
9	StressResource001008	StressResource001008 Resource	Stress	No	SEMI E10 > Standby	StressVendor	StressModel
10	StressResource001009	StressResource001009 Resource	Stress	No	SEMI E10 > Standby	StressVendor	StressModel
11	StressResource001010	StressResource001010 Resource	Stress	No	SEMI E10 > Standby	StressVendor	StressModel
12	StressResource001011	StressResource001011 Resource	Stress	No	SEMI E10 > Standby	StressVendor	StressModel
13	StressResource001012	StressResource001012 Resource	Stress	No	SEMI E10 > Standby	StressVendor	StressModel
14	StressResource001013	StressResource001013 Resource	Stress	No	SEMI E10 > Standby	StressVendor	StressModel
15	StressResourceConsumableFeed00101	StressResourceConsumableFeed001010 Res	Stress	No	SEMI E10 > Standby	StressVendor	StressModel
16	StressResourceConsumableFeed00101	StressResourceConsumableFeed001011 Res	Stress	No	SEMI E10 > Standby	StressVendor	StressModel
17	StressResourceConsumableFeed00101	StressResourceConsumableFeed001012 Res	Stress	No	SEMI E10 > Standby	StressVendor	StressModel
18	StressResourceConsumableFeed00101	StressResourceConsumableFeed001013 Res	Stress	No	SEMI E10 > Standby	StressVendor	StressModel
19	StressResource001014	StressResource001014 Resource	Stress	No	SEMI E10 > Standby	StressVendor	StressModel
20	StressResource001015	StressResource001015 Resource	Stress	No	SEMI E10 > Standby	StressVendor	StressModel
21	StressResource001016	StressResource001016 Resource	Stress	No	SEMI E10 > Standby	StressVendor	StressModel
22	StressResource001017	StressResource001017 Resource	Stress	No	SEMI E10 > Standby	StressVendor	StressModel
23	StressResource001018	StressResource001018 Resource	Stress	No	SEMI E10 > Standby	StressVendor	StressModel
24	StressResource001019	StressResource001019 Resource	Stress	No	SEMI E10 > Standby	StressVendor	StressModel
25	StressResource001020	StressResource001020 Resource	Stress	No	SEMI E10 > Standby	StressVendor	StressModel

... <DM>Resource <DM>Parameter <DM>Protocol ProtocolStates <DM>DataCollection DataCollectionParar ...

Figura 20: Folha do MasterData correspondente aos *Resources*. Cada coluna corresponde a um dos parametros necessários à criação de cada objeto.

5.2.2 Aplicação

Tendo como ponto de partida a aplicação utilizada para carregar os dados para o cmNavigo, a solução mais óbvia seria então adaptar essa própria aplicação para suportar também o *download* dos dados para um ficheiro Master Data vazio. Nas Figuras 21 e 22 é possível ver o resultado final.

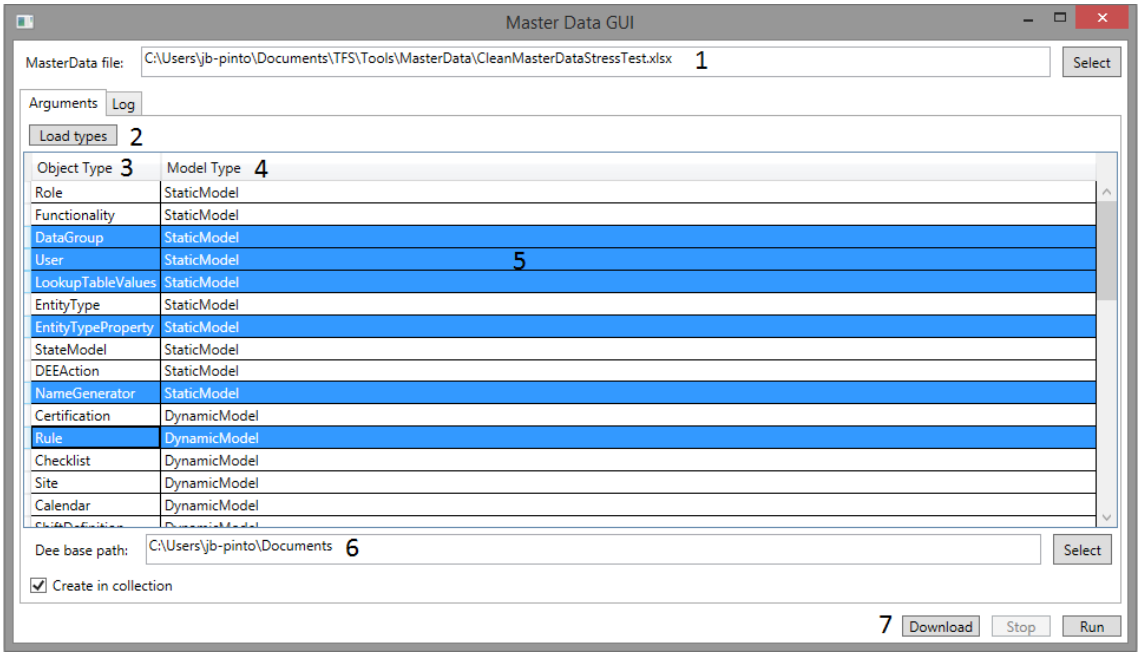


Figura 21: Uma utilização do MasterLoader

A primeira operação que o utilizador executa é selecionar o ficheiro Master Data que irá ser preenchido (1). Caso alguma das folhas já tenha conteúdo este será apagado. Após selecionar o ficheiro, é possível carregar para a aplicação todas as folhas presentes naquele Master Data (2), já que existem diferentes ficheiros Master Data com folhas distintas. Depois de estarem carregadas todas as folhas disponíveis, a Grid é preenchida com o nome do objeto (3) e o tipo do modelo (4). Agora é possível fazer *download* apenas dos objetos que interessam, basta selecioná-los como é possível ver na figura (5). Um dos objetos possíveis tem uma particularidade: tem código-fonte associado. Este objeto é o DEEAction e quando está selecionado, é possível escolher também o caminho onde irão ser guardados os ficheiros que irão conter o código-fonte (6). Por fim, basta clicar em 'Download' (7) para o processo se iniciar.

Assim que a operação de *download* dos dados começa, é possível alterar a vista para a tab Log (1), onde se tem acesso a algumas informações sobre o estado atual do que está a ser descarregado do sistema (2). Por fim, caso seja necessário interromper a operação a meio, existe uma forma simples de o fazer (3).

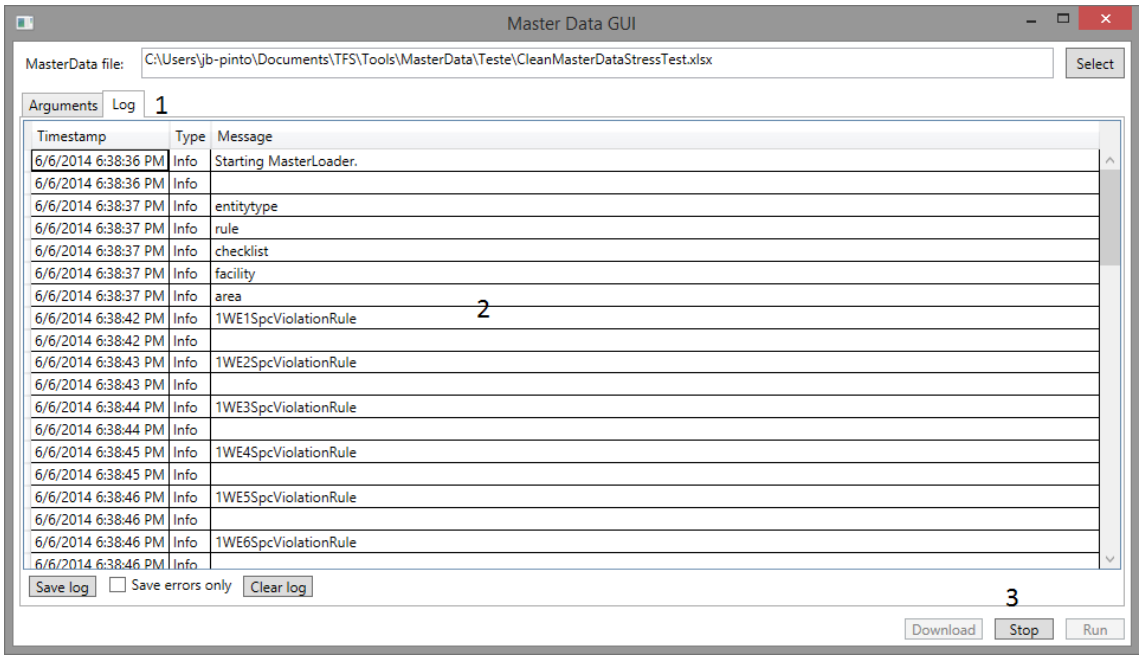


Figura 22: Uma utilização do Master Loader

Através de uma simples adaptação de uma aplicação já existente foi possível cumprir um dos principais requisitos do projeto: conseguir extrair os dados do sistema após a simulação terminar. Neste caso não foi contra-produtivo começar pelo fim, já que seria expectável que a aplicação que fizesse a extração dos dados depois destes serem simulados fosse desenvolvida em último lugar, pois foi possível estudar o cmNavigo de uma forma muito mais prática antes de se partir para a parte a simulação, o que facilitou muito todo o processo.

5.3 cmSimulatorConfig

Como já foi explicado, antes da simulação acontecer existe uma etapa fundamental: a criação do modelo. Em termos práticos, um modelo de simulação não é nada mais do que a esquematização de todas as variáveis, necessárias (obrigatórias) e configuráveis (opcionais), ao correto funcionamento da simulação. Optou-se por separar a criação do modelo da execução da simulação apenas por uma opção logística, já que no fim o modelo será guardado num ficheiro XML que pode ou não ser editado manualmente antes da execução propriamente dita.

No capítulo 3 foram discutidos os vários exemplos de modelos de simulação atualmente existentes. Apesar de a ideia inicial passar por adaptar o modelo apenas a agentes, ficou visível com o estudo do cmNavigo que este já dividia os passos de uma forma que seria facilmente aproveitada: através de eventos. Os *Steps* representam os eventos que um *Material* tem de percorrer, fazendo então sentido que o modelo adotado se baseie na modelação por eventos com

uma *nuance*: cada *Material* pode ser representado como um agente que percorre a linha de eventos que lhe está associada.

5.3.1 Aplicação

Ao conceber a aplicação de configuração, a principal preocupação foi tentar deixar o utilizador personalizar o máximo de informação possível sobre a simulação. Assim, a informação foi separada em quatro abas diferentes: *Products*, *Materials*, *Resources* e *ReworkSteps*.

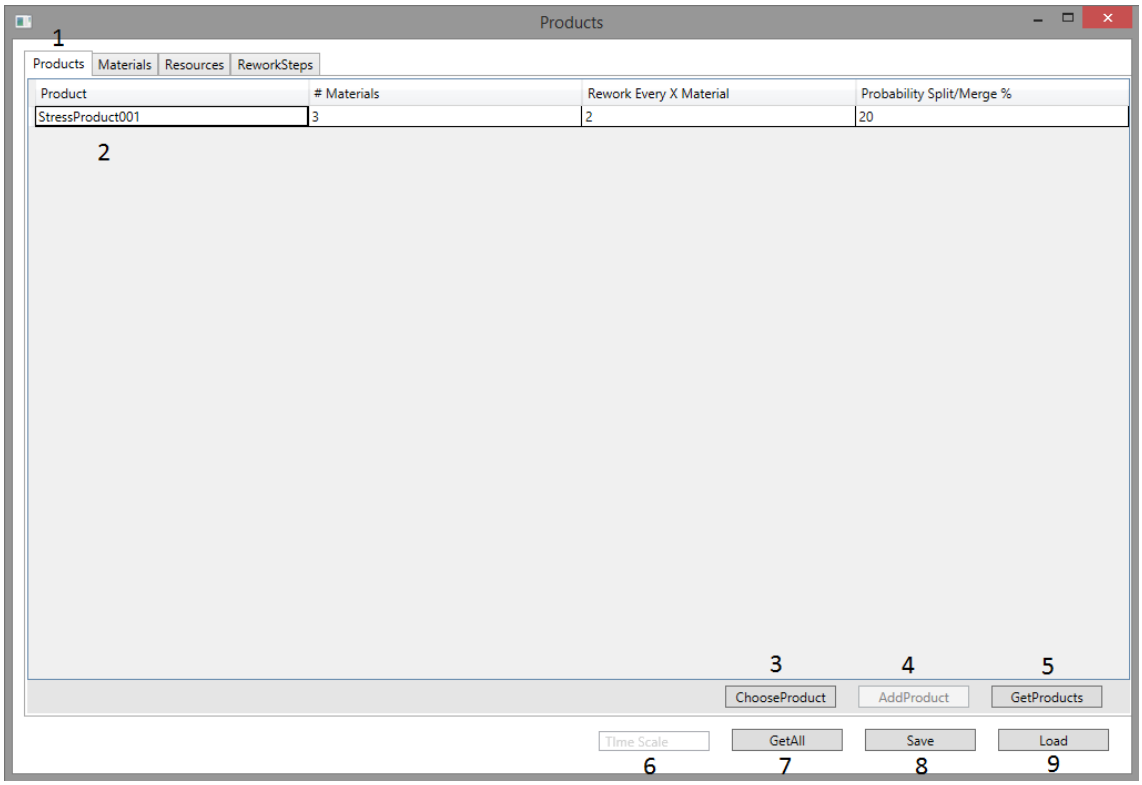


Figura 23: Aba de escolha dos Produtos a serem usados na simulação

A primeira aba (1) pode ser vista na Figura 23 e contém a informação sobre os produtos existentes no cmNavigo e que podem ser simulados. O primeiro passo seria obter todos os produtos existentes diretamente da base de dados (5). Alternativamente pode-se obter toda a informação diretamente da base de dados (7) ou aquela que está guardado num ficheiro XML de uma anterior utilização (9). Ao ir buscar a informação ao ficheiro XML o tempo de processamento é consideravelmente menor, corre-se apenas o risco da informação estar incorreta/desatualizada, daí ser possível atualizar a informação por partes (5) ou toda de uma vez (7). Após este passo, é possível escolher um produto (3), escolhendo também o número de materiais que este vai ter, de quantos em quantos materiais é que irá existir *rework* (esta função é extremamente importante quando o número de materiais escolhido é elevado) e a

probabilidade do material fazer *split* e *merge*. No fim, basta adicionar o produto (4) à *grid* de produtos atualmente existente (2). Existem ainda duas opções que são independentes de qualquer aba: a escolha do tempo para a escala (6), já que durante as ações onde seja necessário respeitar o tempo terá de haver uma escala caso existam tempos muito longos, e o guardar da informação que se está a configurar (8), criando o já falado ficheiro XML.

A aba dos materiais (1), que pode ser visualizada na Figura 24, é preenchida com valores *default* após ter sido adicionado um produto na aba anterior. Neste caso, como se escolheram

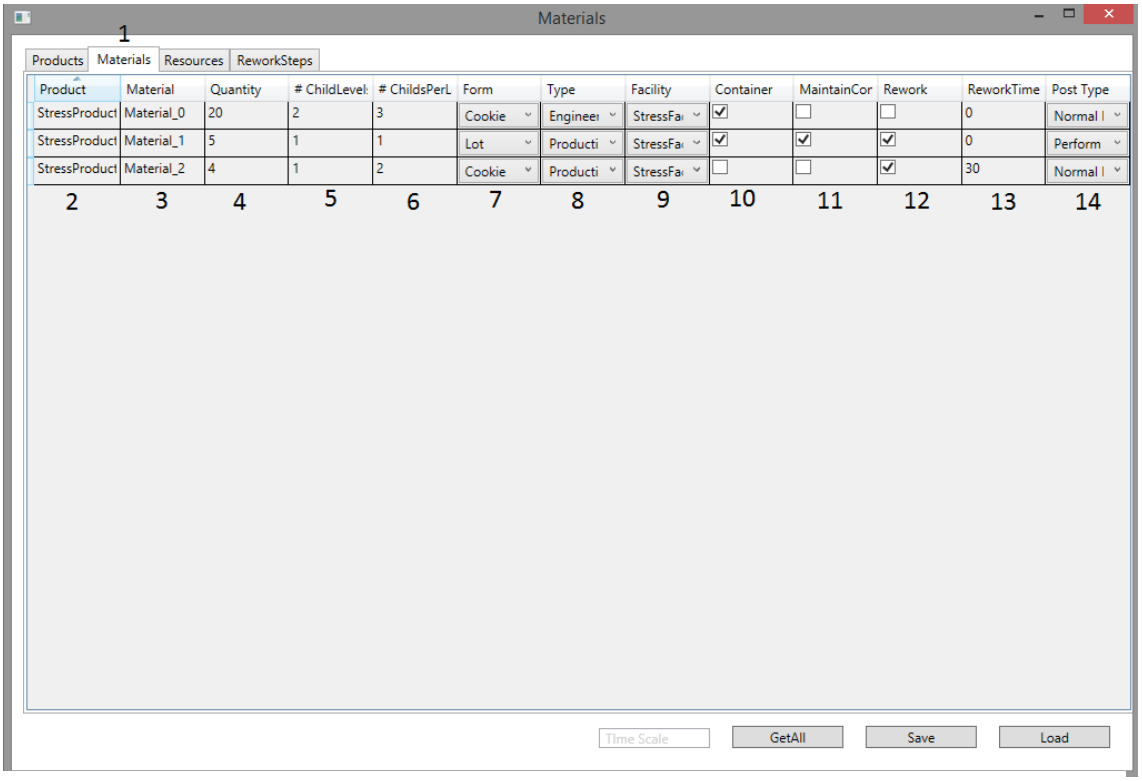


Figura 24: Configuração dos Materials usados na simulação

três materiais para o produto adicionado, foram acrescentados então três materiais configuráveis. O nome do produto (2) é meramente uma informação visual, assim como o nome do material (3) que não podem ser alterados. A quantidade de cada *Material* (4) representa o número de unidades que aquele material vai produzir. Por exemplo, se um material for uma *wafer*, a quantidade representa o número de componentes que aquela *wafer* irá conseguir produzir. O número de niveis (5) e o número de filhos por nível (6) são os indicativos da família que um material pode ter, já que um *Material* pode representar apenas um material como um conjunto destes. A forma (7) e o tipo (8) são duas opções para a caracterização do material. A instalação (9) indica o local onde o *Material* estará inicialmente. Um *Material* pode estar associado a um contentor (10) e além disso pode querer manter esse mesmo container quando muda de instalação (11), senão mudará para um contentor da instalação para onde vai. Em relação ao *rework* dos materiais, é possível escolher se esse material o irá fazer (12) e se esse

rework depende do tempo ou não (13). Por fim, é possível escolher se algumas das operações internas do cmNavigo (*Checklists*, *Data Collections*, ...) irão ser efetuadas separadamente ou todas de uma vez (14).

É também possível configurar algumas opções em relação aos *Resources* (1). Na Figura 25 é possível ver a *grid* que contém todos os *Resources* disponíveis no cmNavigo, identificados pelo nome (2). É possível configurar um recurso para falhar de x em x *TrackIn's* (3), ou seja,

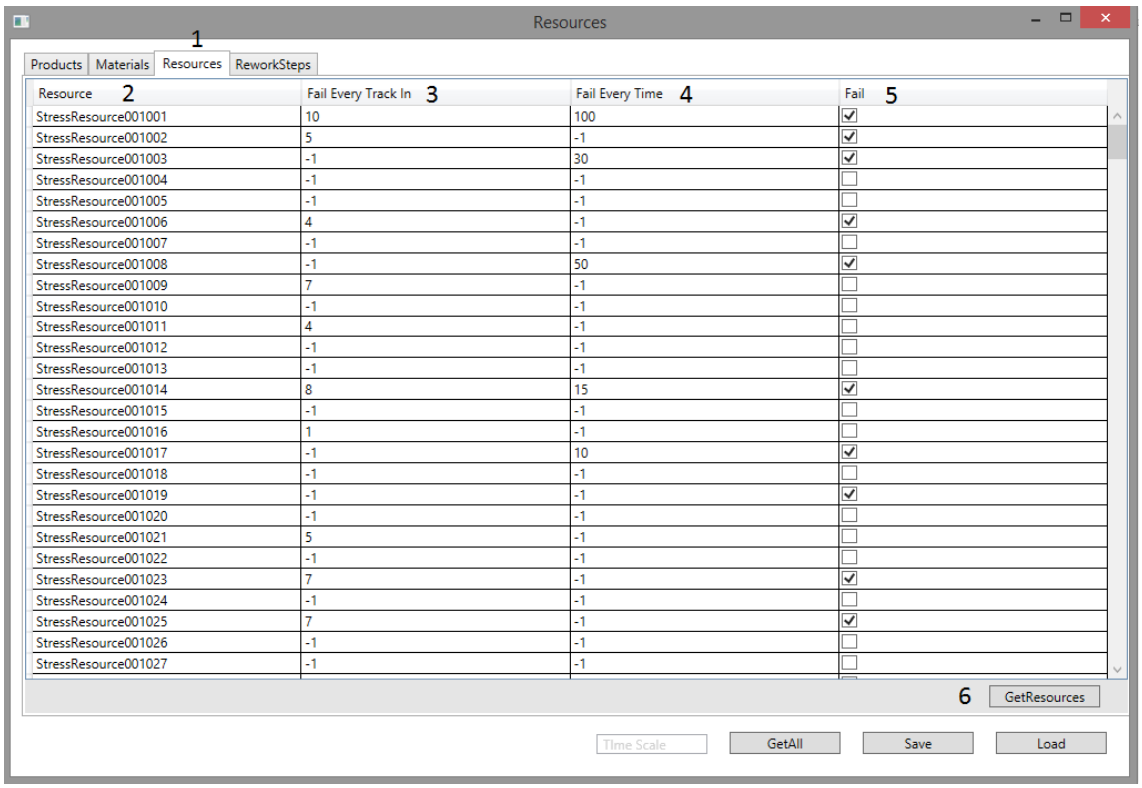


Figura 25: Escolha dos Resources a serem testados

sempre que se fizer um número pré-determinado de TrackIn's, aquele recurso ira estar indisponível. É ainda possível obter o mesmo resultado através de um tempo pré-determinado (4). Por fim, qualquer uma destas configurações só é possível se ativarmos a desativação do resource (5), uma precaução extra. Tal como acontece com os produtos, é também possível atualizar apenas a lista de recursos (6).

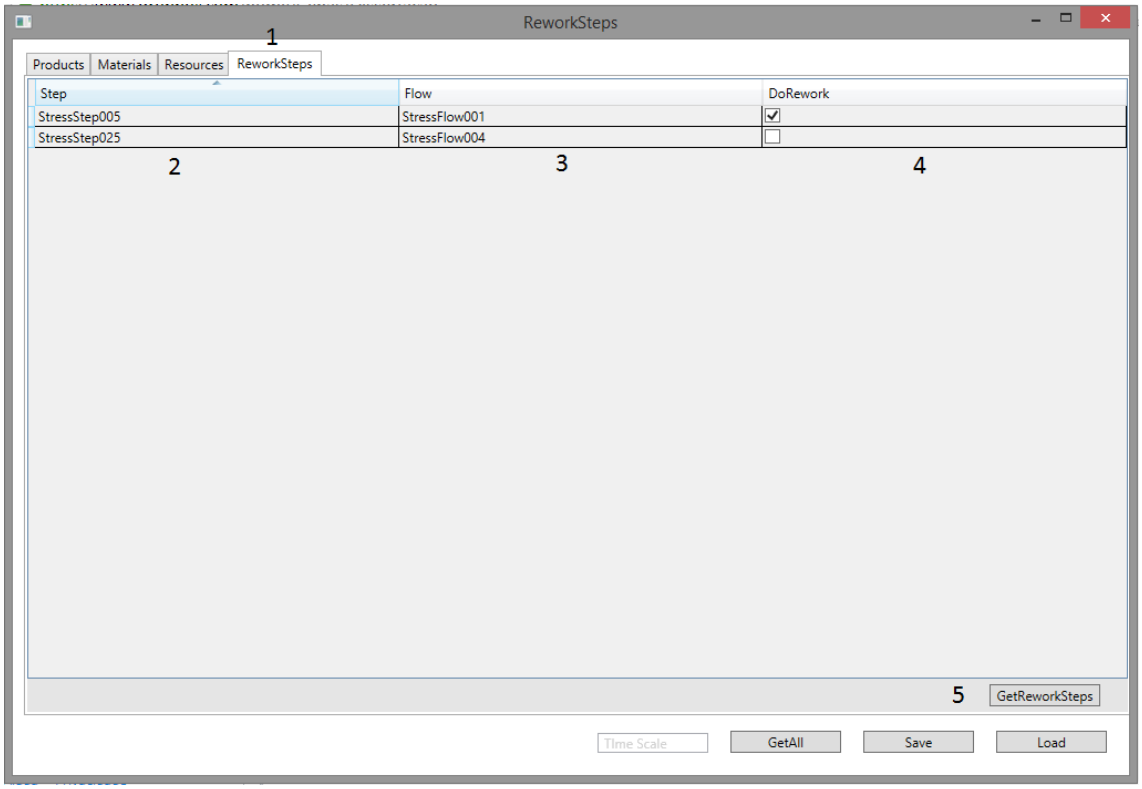


Figura 26: Steps possíveis de um Material efetuar Rework

A última aba corresponde aos *Steps* onde pode existir *Rework* de materiais (1). Na Figura 26 é possível ver a grid preenchida com os *Steps* (2) e respetivos *Flows* (3) onde é possível um material repetir aquele passo. A escolha aqui recai apenas se esse *rework* pode ou não existir (4). Mais uma vez, esta *grid* pode ser atualizada para a versão mais recente independentemente das outras (5).

Ainda dentro da aplicação, é possível falar sobre o ficheiro XML que é criado e pode ser posteriormente importado/modificado. O objetivo principal passou por criar um ficheiro facilmente legível para poder ser modificado manualmente. A informação que é guardada é então:

- *Products*, onde se guarda o nome do produto e o número de materiais que tem associado. Caso esse produto ainda não tenh sido usado, o número de materiais será -1;
- *Containers*, esta informação é guardada aqui não por ser diretamente possível a sua customização, mas porque a simulação irá precisar desta informação e assim é mais um pedido ao servidor que não é necessário repetir;
- *Materials*, juntamento com toda a informação que foi discutida na aba de materiais;
- *Resources*, seguindo o mesmo processo dos materiais;
- *ReworkSteps*, também com a informação presente na respetiva aba.

Guardando apenas esta informação, e nem sendo necessário contar com os *Containers*, é possível carregar este ficheiro para efetuar algum tipo de modificação e guardá-lo novamente ou então partir diretamente para a simulação.

A mais-valia de ter este tipo de aplicação separada do simulador é principalmente uma questão de poupança de recursos. Para ser possível escalar o simulador por várias máquinas, este tem de ser o mais eficiente em termos de recursos possível. Assim, se o modelo for criado separadamente, a aplicação do simulador apenas tem de tratar de carregar os dados já configurados e simulá-los.

Em relação ao *output* da aplicação, a criação de um ficheiro XML e não de outro tipo mais eficiente é essencial caso seja necessário alterar algum parâmetro diretamente no modelo, sem ser necessário recorrer à aplicação.

5.4 cmSimulator

Tendo o modelo criado, falta então concretizar a última etapa de todo o processo: a simulação. Já foi explicado no capítulo anterior o que significa a simulação de um MES, especificamente do cmNavigo, e o cmSimulator faz exatamente isso. Fazendo uso da *framework* WCF, é possível utilizar serviços internos do cmNavigo para simular os processos reais de uma fábrica.

A aplicação tem uma interface muito simples, pois a simulação está a acontecer diretamente no cmNavigo. Através deste é possível visualizar o que cada *Material* está a fazer em determinado momento. Foi aproveitado um módulo já existente, o fabLive, para conseguir acompanhar a simulação de uma forma mais visual, eliminando esta parte do simulador e tornando-o mais eficiente.

5.4.1 Lógica

A lógica que se esconde por detrás da simulação começa na forma como a aplicação está estruturada. Após importar o ficheiro XML criado pelo cmSimulatorConfig e começar a simulação, o modelo é criado. Os *Materials* que irão ser usados na simulação são efetivamente criados no cmNavigo e cada *Material* é lançado num *thread* diferente, tentando assim que cada um seja representado o mais perto possível de um agente. Cada *thread* irá então processar os *Steps* que estão associados àquele *Material*, interagindo com os outros *Materials* quando é necessário.

A simulação mais simples que pode acontecer é o deslocamento de um *Material* pela fábrica. Para isso, existem sempre cinco operações básicas que serão sempre executadas em qualquer simulação.

- ***Dispatch***: Fazer *Dispatch* de um *Material* não é nada mais do que colocá-lo pronto para ser processado por um *Resource*, reservando ao mesmo tempo esse mesmo *Resource*. As alterações que são feitas neste passo são apenas no estado do *Material*.
- ***Track In***: *Track In* representa a ação de colocar um *Material*, no *Resource* que foi escolhido anteriormente, a ser processado. Numa situação real, esta operação pode ser efetuada automaticamente por uma máquina ou manualmente por um operador no local. Neste caso as alterações já se refletem também no *Resource* anteriormente escolhido.
- ***Track Out***: A operação inversa da anterior. Efetuar um *Track Out* significa que o *Material* já acabou de ser processado e está pronto para o próximo passo do *Flow*.
- ***Move Next***: Ao efetuar *Move Next*, o simulador seleciona o próximo *Step* do *Flow* que o *Material* está a cumprir e transporta-o para aí. O que significa isto em termos práticos é que após fazer *Move Next* existem duas opções: ou este ciclo se repete ou então, caso o *Step* atual seja o último, o *Material* é terminado e a *thread* da simulação também termina.
- ***Terminate***: A operação que termina o *Material* acontece apenas no caso anteriormente descrito ou caso exista uma falha de *Resources* em algum ponto da simulação.

Além da simulação base, foram implementadas uma série de operações que são possíveis de incluir na simulação. O objetivo é que o utilizador consiga testar o máximo de funcionalidades possíveis do cmNavigo. Algumas das operações já foram descritas superficialmente no capítulo 5.3.1 aquando da descrição da sua personalização. De seguida são então apresentadas todas as operações implementadas, personalizáveis ou não.

5.4.1.1 Associação de *Containers*

Um *Container* no contexto do cmNavigo é um contentor que pode transportar *Materials* pela fábrica e entre diferentes instalações. É possível associar um contentor a um *Material* e é possível manter esse mesmo contentor caso o *Material* mude de instalações, já que normalmente um contentor pertence apenas a uma instalação.

O que acontece na simulação é que é atribuído um contentor aleatório que esteja disponível ao *Material* e, caso este mude de instalação, a opção que foi selecionada mantém-se: ou o contentor é desassociado do *Material* e é procurado um *Container* novo na instalação para a qual vai o *Material* ou o contentor é transportado juntamente com o *Material*.

5.4.1.2 Rework de *Materials*

Por vezes numa linha de produção é necessário voltar a processar matéria-prima por diferentes motivos, ou seja repetir o processo que se acabou de fazer. No cmNavigo essa opção tem o nome de *Rework* e pode ser efetuada por qualquer *Material*, num determinado *Step* que contenha essa opção.

Na simulação, o serviço que faz *Rework* do *Material* é chamado um número de vezes pré-definido. Seria possível adicionar uma variável à configuração e personalizar também essa parte, mas não é fundamental. Quando o *Material* terminar então de fazer *Rework*, a simulação continua normalmente.

5.4.1.3 Falha de *Resources*

Algo normal e constante é a falha de uma máquina numa linha de produção, sendo algo com que se tem de lidar a nível diário. Torna-se então importante simular que uma determinada máquina falha num determinado espaço de tempo ou depois de um determinado número de utilizações.

Na simulação o que acontece é que quando um dos Resources que está programado para falhar atingi essa condição, o seu estado é alterado para *Unschedule Down* (está numa paragem não programada) antes de um *Track In* e assim o *Material* não o pode escolher para efetuar determinado *Step*.

5.4.1.4 Yield and Cycle Time

Como é normal, cada *Resource* tem um tempo constante que demora a processar um *Material*, ou seja, o tempo que vai entre o *Track In* e o *Track Out* desse Material. Como o objetivo da simulação é que esta seja a mais próxima da realidade possível, esses tempos têm que ser respeitados.

Durante a simulação é então verificado o tempo real que essa operação iria demorar sendo depois convertido para uma escala. Isto é extremamente importante pois se os tempos excedessem os minutos, isso iria significar que a simulação iria demorar todo esse tempo também. Convertendo para uma escala permite agilizar todo o processo.

5.4.1.5 *Data Collections*

As *Data Collections* serão provavelmente uma das mais importantes partes de um MES. Recolher dados de performance para estes serem analisados posteriormente é uma das principais formas de conseguir perceber o que está a funcionar bem e mal, onde e quando. No cmNavigo, estes dados podem ser guardados em quatro diferentes pontos da transformação de um *Material*: *Track In*, *Track Out*, *Rework* e *Move Next*.

O que acontece na simulação é que durante as quatro operações mencionadas acima, é efetuada uma verificação para saber se naquele ponto é necessário realizar uma recolha de dados. Se sim, esses dados serão recolhidos de acordo com o que o utilizador especificou na configuração do modelo: recolher os dados um a um e anotá-los no cmNavigo ou guardá-los todos de uma vez e enviá-los só no fim.

5.4.1.6 Família de *Materials*

Como já foi referido, um *Material* pode também representar um conjunto de *Materials*. Este tipo de Material específico tem por nome família de materiais. Quando se cria um *Material*, existem dois parâmetros que podem ser preenchidos em relação à sua família: profundidade dos níveis e número de filhos por nível. Cada *Material* filho é exatamente igual ao pai em todos os aspetos.

Sempre que é feito um *Track In* e um *Track Out* durante a simulação, é necessário especificar essas mesmas operações para cada um dos filhos. Todas as outras operações são feitas em conjunto.

5.4.1.7 *Splits*/*Merges*

O conceito de *Split* e de *Merge* é mais facilmente explicado com um exemplo. Um dos *Materials* mais utilizados pelo cmNavigo é a wafer. Uma wafer é um disco com um número variado de microcircuitos que depois de processada gera microprocessadores. Para ser corretamente processada em algum passo, pode ter que ser dividida. O processo de dividir um *Material* criando um novo chama-se *Split*. O que acontece na realidade é que o *Material* original é reduzido em determinada quantidade, quantidade esta que passa a estar incluída no novo *Material* criado. *Merge* é o processo inverso, juntar dois ou mais *Materials* que foram separados.

Quando uma percentagem maior do que zero é selecionada na configuração do modelo, o que acontece durante a simulação é que, caso essa probabilidade seja atingida e o Material tenha quantidade suficiente para se dividir em dois, acontece um *Split*. Em relação ao *Merge*, só pode acontecer com *Materials* que já tenham sido anteriormente divididos. Quando um *Material* que já foi dividido é selecionado para fazer *Merge*, o que acontece é que irá ser juntado com todos os *Materials* que foram divididos, assim como os que foram divididos a partir destes, voltando a ter a quantidade inicial.

5.4.1.8 Checklists

Por vezes existe uma série de ações, sejam manuais ou automáticas, que precisam de ser realizadas em alguns passos da transformação do *Material*. Para isso o cmNavigo permite criar

Checklists, que tal como o nome indica, nada mais são que o conjunto dessas mesmas ações. Mais uma vez, as ações podem acontecer nas já faladas quatro operações das Data Collections: *Track In*, *Track Out*, *Rework* e *Move Next*.

Como seria de esperar, o processo é semelhante. Primeiro é verificado se será necessário efetuar uma *Checklist* e se sim, efetuar as ações uma a uma ou todas de uma vez.

5.4.1.9 *Maintenance Management*

Um *Resource* tem vários estados possíveis onde não está a funcionar: *Nonscheduled* (não está programado para funcionar), *Schedule Down* (está numa paragem programada) e *Unscheduled Down*. Quando está numa paragem programada, existe um plano de manutenção atribuído que irá ser executado a seu devido tempo. Mas enquanto no primeiro e no segundo caso o *Resource* está assim por opção, no terceiro não. É então necessário haver uma intervenção imediata. É este o papel da *Maintenance Management*, assegurar que os planos de manutenção estão a ser executados quando são devidos.

O que irá acontecer em termos de simulação será que em cada *Step*, um dos *Resources* é escolhido aleatoriamente e é-lhe aplicado um plano de manutenção. O que se irá verificar é que o *Resource* escolhido será manualmente configurado para estar em baixo para ser possível aplicar-lhe o plano. No fim o *Resource* tem que estar disponível de novo para ser usado. Durante todo este processo, não pode ser usado por nenhum dos *Materials* da simulação.

5.4.2 Aplicação

Como já foi explicado, o objetivo será minimizar os recursos gastos pela aplicação, concentrando-os na simulação. Assim, a interface do simulador limita-se a permitir que o utilizador escolha o ficheiro XML onde está o modelo e a correr a simulação. Durante a simulação, o simulador apenas tem a indicação dos *Materials* que estão em curso, onde se pode saber se estes já terminaram a execução ou se ainda estão em trânsito, como se pode ver na Figura 27.

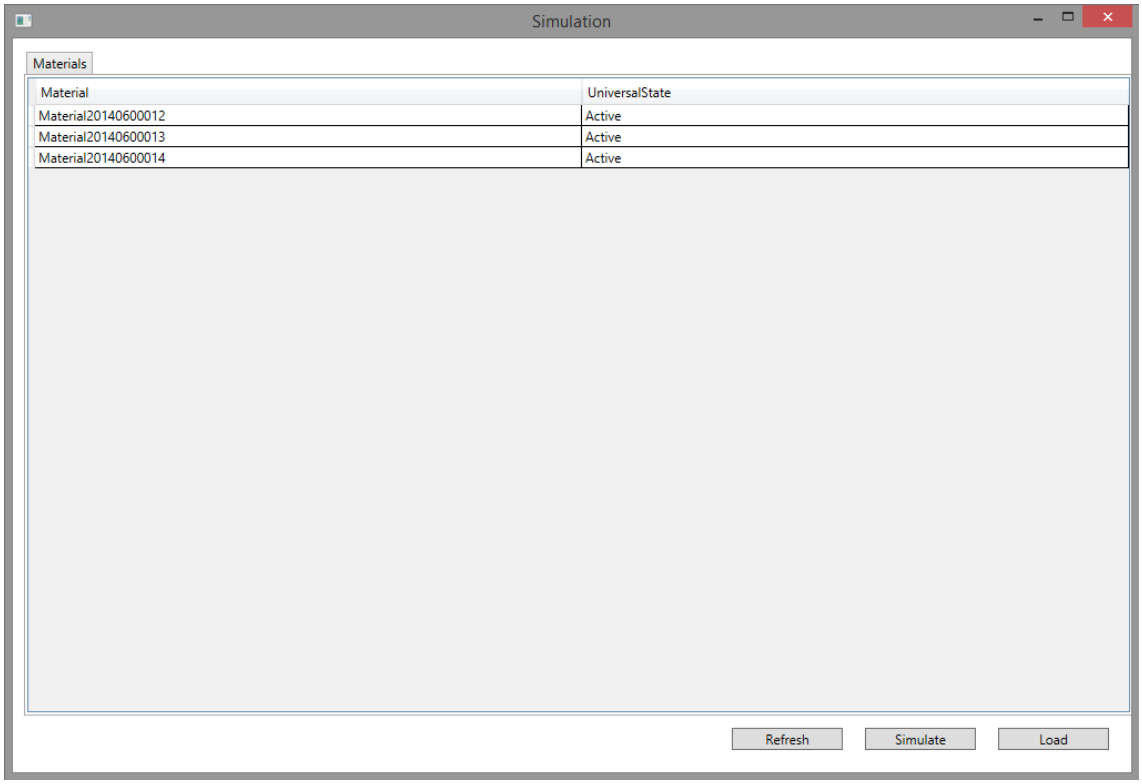


Figura 27: Interface do cmSimulator

A simulação pode então ser visualizada com mais detalhe diretamente no cmNavigo. Existem duas formas de o fazer: criar um fabLive com todos os *Resources* disponíveis e visualizar algumas das operações que são possíveis de controlar desta maneira (Figura 28) ou,

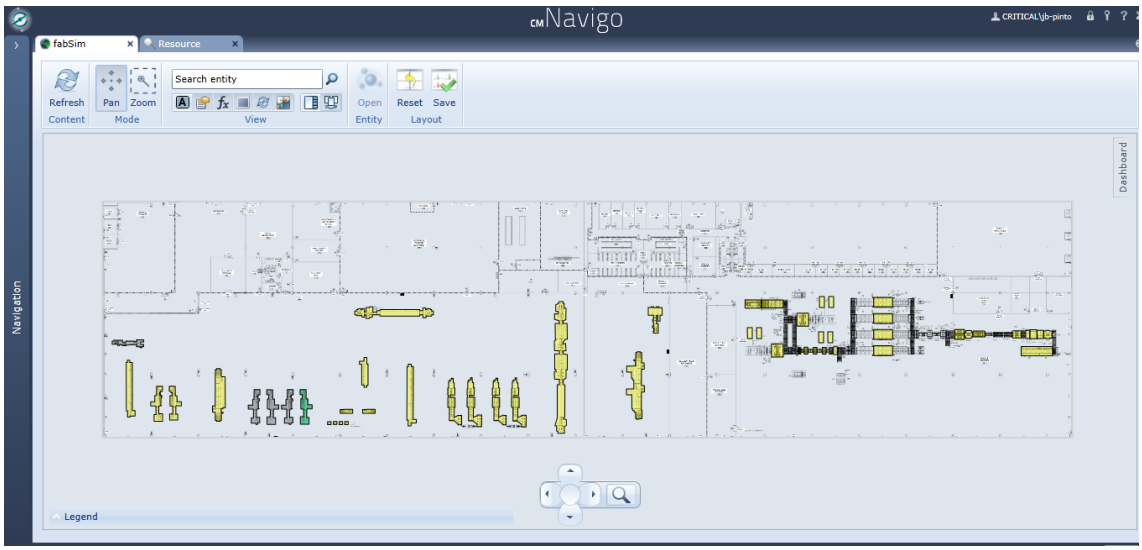


Figura 28: Visualização da Falha de Resources no fabLive. Os Resources a amarelo estão funcionais, o verde está a trabalhar num Material neste momento e os cinzento estão em baixo.

caso a operação não se consiga ver desta forma, consultar diretamente o histórico do *Material* (Figura 29) onde qualquer das operações que são efetuadas estão descritas pormenorizadamente.

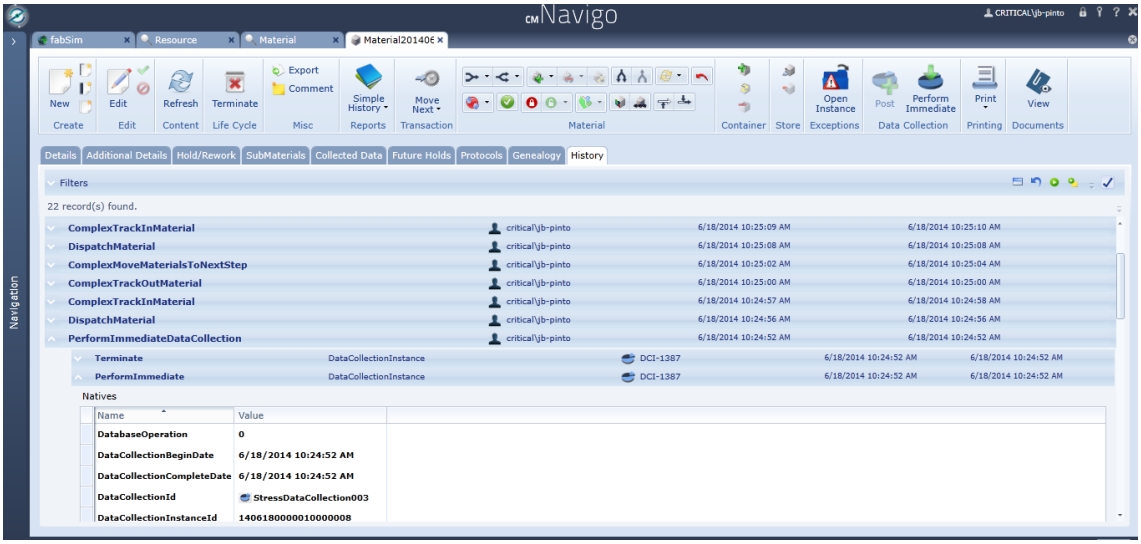


Figura 29: Histórico detalhado de um Material.

Como seria expectável, esta foi a parte mais demorada do projeto. Simular várias operações sobre vários *Materials*, ao mesmo tempo que se tenta manter a simulação o mais eficiente possível, é um desafio imenso. Apesar de tudo, as operações que foram descritas em cima foram implementadas com sucesso, existindo apenas um senão numa das operações: a Falha de *Resources* não pode ser implementada em várias máquinas por causa de erros de concorrência. O cmNavigo é uma ferramenta que foi desenvolvida pensando que quem a utilizasse, seria diretamente na interface e não através dos serviços WCF que disponibiliza, serviços estes que até agora eram usados maioritariamente para testes. Por causa disto, é impossível saber quando outra máquina está também a utilizar o cmSimulator. Apesar de ter sido desenvolvida uma solução local para resolver o problema da concorrência (vários materiais acederem ao mesmo serviço) através da implementação de agentes, quando a Falha de *Resources* está ativa as outras máquinas não o conseguem saber e podem usar *Resources* que num determinado momento estão ativos mas, porque na outra máquina esses *Resources* estão a ser alterados, no momento a seguir já não o estão. Além deste senão, o resto das funcionalidades foram implementadas com sucesso a 100%.

5.5 Sumário

Através da utilização destas três aplicações tornou-se possível simular a utilização do cmNavigo numa fábrica. Apesar de não terem sido implementadas todas as funcionalidades possíveis pois o intervalo de tempo para a realização do projeto não era longo, foram implementadas as que a equipa considerou essenciais. No fim, cerca de 80% das funcionalidades do cmNavigo ficaram disponíveis para serem simuladas.

O primeiro passo será sempre carregar um ficheiro de Master Data para a ferramenta. Este ficheiro representa todas as variáveis que representam uma fábrica, desde a sua linha de produção aos seus planos. Caso seja necessário acrescentar ou modificar algo da simulação, será sempre possível fazê-lo diretamente através da interface do cmNavigo. Quando os dados estiverem corretamente preenchidos, é possível então criar o modelo que irá ser simulado. Para isso basta criar os *Materials* que se querem ver simulados, personalizando todas as suas variáveis. Quando esta fase estiver concluída, mais uma vez é possível alterar o modelo diretamente no ficheiro XML sem ser necessário recorrer à aplicação. A fase da simulação é extremamente simples já que não envolve *inputs* por parte do utilizador, basta para isso carregar o modelo e esperar que esta termine. É possível acompanhar ao mesmo tempo que esta acontece no cmNavigo através de um fabLive ou do histórico do *Material*. No fim, caso seja necessário, os resultados da simulação podem ser extraídos para criar um novo ficheiro de Master Data, possibilitando a inserção dos dados simulados noutro sistema diferente.

Capítulo 6

Resultados e Discussão

Neste capítulo são apresentados os principais resultados obtidos com a realização deste projeto juntamente com uma breve discussão sobre estes. De seguida é apresentada a validação e verificação que foi efetuada para que este projeto fosse corretamente desenvolvido. No fim são apresentadas as principais conclusões retiradas e é indicado o caminho a seguir caso haja continuação deste projeto.

Ao contrário de um projeto de investigação, os resultados de um projeto maioritariamente prático são focados essencialmente no sucesso ou no insucesso da aplicação (ou aplicações). Neste caso é possível ainda retirar resultados da implementação do modelo de simulação novo que combina Eventos Discretos com Agentes, sendo estes exclusivamente sobre a utilidade do mesmo.

A grande diferença entre um modelo que usa apenas eventos para a sua modelação e um que alia também agentes é a possibilidade de modelar situações que não estão pré-definidas à partida. Modelar apenas eventos discretos funciona perfeitamente quando temos uma série de operações pré-definidas que têm que ser executadas numa determinada ordem, tal como acontece com os *Steps* no cmNavigo. Quando se quer simular uma situação que implique algum tipo de decisão por parte do simulador, isto é, informação que não esteja disponível à partida mas que foi descoberta durante a simulação, um evento não basta. A introdução de agentes colmata esta falha na simulação de eventos. Neste caso específico, a introdução de agentes foi extremamente útil no já explicado caso da Falha de *Resources*. Caso se optasse por manter apenas os eventos, teria que ser implementada uma máquina de estados que tentasse adivinhar todos os resultados possíveis. Além de ser uma solução que implica gastar mais recursos e mais tempo, nunca será uma solução que cubra todos os resultados como um agente que se adapta às situações, daí a utilidade de se combinarem estas duas abordagens neste projeto.

De seguida é apresentada uma tabela com os resultados das várias simulações efetuadas, onde se mostra o número de testes efetuados por cada operação implementada assim como o sucesso ou não da mesma.

Tabela 2: Resultados da Simulação

Funcionalidade	Número de Testes	Totalmente Funcional
Simulação Base	500+	Sim
Associação de *Containers*	50+	Não
Rework de *Materials*	30+	Sim
Falha de *Resources*	85+	Não
Yield and Cycle Time	15+	Sim
Data Collections	70+	Sim
Família de *Materials*	25+	Sim
Splits	100+	Sim
Merge	75+	Sim
Checklists	70+	Sim
Maintenance Management	30+	Sim

A disparidade do número de testes é simples de explicar. A simulação base participou em todos os testes. As operações têm um número de testes diferentes consoante a sua dificuldade de implementação. Efetuar um *Split* a um *Material* implica lançar uma nova *thread* (e consequente novo agente) para que este se torne independente e faça uma simulação completamente separada do *Material* pai, enquanto numa *Yield and Cycle Time* basta verificar o tempo que aquele *Resource* precisa para processar um *Material* e implementar um *delay* que foi previamente transformado para uma escala.

As duas únicas funcionalidades que não estão implementadas a 100% são a Falha de *Resources*, já explicada no capítulo anterior, e a Associação de *Containers*. Esta última não ficou completamente implementada devido a outro bug do cmNavigo, sendo que este já era conhecido. O que acontecia neste caso é que não permitia que dois *Materials* do mesmo tipo fossem adicionados ao mesmo *Container*, apesar de isso na realidade ser possível. Assim, quando esse bug ocorria a execução da simulação terminava. Quando o *bug* não ocorria a operação funcionava corretamente.

6.1 Validação e Verificação

Os métodos para efetuar a verificação e a validação do sistema já foram discutidos no capítulo 4.3. Além de todos os passos ditos normais para a verificação, existiu ainda um passo adicional: a verificação diretamente no cmNavigo das operações que estavam a ser efetuadas pela simulação. Assim, qualquer operação que estava a ser implementada era seguida de algumas execuções da mesma lógica diretamente na ferramenta.

Através desta verificação pormenorizada, foi até possível descobrir um bug no cmNavigo. Um *Resource* pode receber vários *Materials* ao mesmo tempo, ou seja, pode receber vários *Track In's* sem ter que fazer necessariamente um *Track Out*. Quando acontece um *Track In*, o estado do *Resource* muda, passando de *Standby* para *Productive*. O que acontece no cmNavigo é que o segundo *Material* que entre no *Resource* tentava mudar o estado do *Resource* de *Standby* para *Productive*, apesar de este já estar no estado *Productive*. A máquina de estados dos *Resources* não estava preparada para lidar com esta situação (*Productive* para *Productive*), já que o cmNavigo era até à implementação do simulador apenas usado na sua interface e este caso particular lá nunca acontecia.

Quanto à validação, esta foi efetuada em dois passos distintos:

- Usando o método descrito em cima. Ao verificar se as operações estavam a ser corretamente executadas, a validação estava a acontecer simultaneamente.
- Através do *feedback* da equipa do produto. Quando uma solução é desenvolvida para um cliente específico, esta só estará totalmente validada quando este estiver satisfeito com o resultado. Ao efetuar *demos* quinzenalmente e diretamente à equipa que irá trabalhar com o simulador, foi possível recolher as várias opiniões e saber em que direção levar o projeto.

A primeira validação foi então feita através dos já falados testes. Inicialmente eram usados exemplos simples apenas para testar se aquela operação específica estava funcional. No fim foi efetuada uma validação final onde foi então criada uma fábrica virtual onde todas as funcionalidades eram usadas. De seguida é apresentada uma enumeração dos dados que compõe a fábrica.

- 2 *Facilites*;
- 4 *Areas*;
- 5 *Checklists*;
- 1 *Calendar*;
- 38 *Services*;
- 30 *Steps*;

- 7 *Flows*;
- 2 *Products*;
- 420 *Resources*;
- 10 *Data Collections*;
- 5 *Recipes*;
- 1 *Maintenance Plan*;
- 11 *Containers*.

O número de *Materials* que foram criados durante a simulação era, até à data da escrita deste relatório, 1158.

Caso fossem utilizados dados de uma empresa real para a validação final, uma ou mais operações podiam não ser incluídas, já que uma fábrica não tem que necessariamente incluir, por exemplo, Associação de *Containers* no seu processo de fabrico. Assim, ao criar uma empresa virtual, foi certificado que todos os passos da simulação eram testados.

A validação que foi efetuada pela equipa do produto, e consequente cliente final, focou-se principalmente na usabilidade da aplicação e na pertinência, utilidade e funcionalidade das operações implementadas.

Em relação à usabilidade foram demonstradas as aplicações a alguns membros e foi-lhes a seguir pedido que as utilizassem. O que se pôde verificar com estes testes foi que a única aplicação que não era por vezes totalmente clara era a aplicação do modelo.

Quanto às operações que foram implementadas, com as demonstrações que foram efetuadas e através da verificação diretamente no cmNavigo, a equipa conseguiu verificar e validar que estas estavam efetivamente a funcionar corretamente.

Capítulo 7

Conclusão e Trabalho Futuro

Aliar a simulação a um Sistema para a Execução da Manufatura parece ser uma tendência a seguir nos próximos anos. Uma empresa que possua uma ferramenta tão poderosa irá definitivamente conseguir destacar-se da restante concorrência.

Apesar de nove das onze principais operações terem sido implementadas com sucesso e das outras duas estarem praticamente concluídas, existem ainda alguns pontos que poderiam ser melhorados, que não foram possíveis de implementar por causa do curto tempo em que a dissertação foi realizada. Como já foi referido, ficaram 20% das funcionalidades do cmNavigo por implementar. Apesar de serem funcionalidades pouco utilizadas ou que ainda não estão disponíveis para os clientes, o simulador não estaria disponível sem estas. Na demo final foram então sugeridas algumas possibilidades:

- Acrescentar as operações que ainda não estão implementadas;
- Possibilitar que a Falha de *Resources* seja possível de ser executada em computadores diferentes;
- Permitir ao utilizador configurar ainda mais parâmetros na configuração no modelo. Um exemplo deste ponto seria retirar a aleatoriedade com que se escolhem *Resources*, *Containers*, *Checklists*, … e permitir que o utilizador tenha acesso à informação do sistema para conseguir selecionar exatamente onde e o que é que o *Material* vai utilizar;
- Melhorar a interface gráfica. Neste momento a interface existente está direcionada principalmente para a eficiência da configuração e da simulação, não tendo havido muita preocupação com o aspeto final;
- Criação de uma aplicação externa que permita controlar simuladores que estão a trabalhar em diferentes máquinas. Através de mensagens multi-cast, poderia ser possível informar todos os simuladores para começarem ou pararem as respetivas simulações. Ou até selecionar apenas um ou mais para o efeito.

- Melhorar a *performance* da simulação. Um dos problemas que ocupou mais tempo durante a implementação foi a *performance* da mesma. Como qualquer operação precisa de ser executada dentro do cmNavigo, a utilização dos *endpoints* WCF é necessário em qualquer ponto da simulação. O grande problema ao comunicar dados a uma ferramenta externa para esta os processar é o atraso que resulta disso. O tempo da simulação está sempre dependente da rapidez com que o cmNavigo executa as operações e comunica o resultado. Esta não é uma funcionalidade que está otimizada pois como já foi referido, até agora tinha sido usada maioritariamente para testes de validação, onde o tempo de execução não era prioritário.

A grande discussão que este projeto gera é sobre a utilidade ou não de se aliar agentes às simulações onde predominam os eventos, tal como foi este caso. Uma simulação que modele os dados apenas para estes serem utilizados por eventos é ao mesmo tempo uma simulação mais simples, que usa menos recurso mas que é ao mesmo tempo muito menos poderosa. Os agentes vieram incluir um método capaz de simular teoricamente qualquer cenário, mas que peca pela dificuldade de implementação e pela quantidade de recursos gastos. Cada paradigma que se inclua no modelo torna a simulação muito mais eficaz, capaz de simular muitos mais cenários, por isso a melhor solução passa por englobar vários paradigmas, tentando extrair o melhor de cada um. Aproveitar os eventos já disponíveis dentro do cmNavigo (*Steps*) e transformar os *Materials* em agentes foi a opção mais viável neste caso, já que com esta combinação foi possível simular todas as operações com sucesso.

Referências

[AIG77] Aigner, D. (1977). "Formulation and Estimation of Stochastic Frontier Production Function Models." Journal of Econometrics 6: 17.

[ANY14] AnyLogic (2014). "Multimethod Simulation Approach." Retrieved 11/02/2014, from http://www.anylogic.com/multimethod-modeling/.

[BRN98] Bolton, R. N. (1998). A Dynamic Model of the Duration of the Customer's Relationship with a Continuous Service Provider: The Role of Satisfaction, Institute for Operations Research and the Management Sciences.

[BAF04] Borshchev, A. and A. Filippov (2004). From System Dynamics and Discrete Event to Practical Agent Based Modeling: Reasons, Techniques, Tools. The 22nd International Conference of the System Dynamics Society. Oxford, England.

[BRA99] Bosch, R. A. (1999). "Integer Programming and Conway's Game of Life." Society for Industrial and Applied Mathematics 41.

[CJS05] Carson, J. S. (2005). Introduction to Modeling and Simulation. Winter Simulation Conference.

[WKV10] Chan, W. K. V., et al. (2010). Agent-Based Simulation Tutorial - Simulation of Emergent Behavior and Differences Between Agent-Based Simulation and Discrete-Event Simulation. Winter Simulation Conference.

[CHY11] Charalabidis, Y., et al. (2011). Enhancing Participative Policy Making Thrugh Modelling and Simulation: A State of the Art Review. European, Mediterranean & Middle Eastern Conference on Information Systems.

[CMW05] Cheong, C. and M. Winikoff (2005). "Hermes: A Methodology for Goal-Orientede Agent Interactions."

[CHD08] Cheong, D. (2008). Goal-oriented Interactions for Intelligent Agents. B.App.Sc. Computer Science. Melbourne, Victoria, Australia, RMIT University. Doctor of Philosophy: 260.

[BMD98] Chopard, B. and M. Droz (1998). Cellular Automata Modeling of Psysical Systems, University of Geneva.

[COS14] Corporation, S. (2014). "SIMUL8 Documentation." Retrieved 11/02/2014, from http://www.simul8.com/products/features/documentation.htm.

[TAS98] De Toni, A. and S. Tonchia (1998). "Manufacturing flexibility: A literature review." International Journal of Production Research 36(6): 1587-1617.

[ELR13] Elliott, R. (2013). Manufacturing Execution System (MES): An Examination of Implementa. San Luis Obispo, Faculty of California Polytechnic State University.

[FRA08] Fernandes, R. A. d. C. (2008). Simulador de Sistemas de Produção e de Informação Industriais. Major Automação, Faculdade de Engenharia da Universidade do Porto.

[FPA95] Fishwick, P. A. (1995). "Simulation Model Design and Execution." Prentice Hall Inc.

[GOD07] Goldsman, D. (2007). "Introduction to Simulation." Proceedings of the 2007 Winter Simulation Conference.

[GOD10] Goldsman, D., et al. (2010). "A brief history of simulation revisited." Proceedings of the 2010 Winter Simulation Conference.

[GGC86] Goodwin, G. C., et al. (1986). "Rapprochement between Continuous and Discrete Model Reference Adaptive Control." International Federation of Automatic Control 22(2): 9.

[HBP12] Hemant B. Patil, P. A. P., Nitin G.Bagul (2012). "Paradigms of simulation - A review." World Journal of Science and Technology 2012.

[IRG08] Ingalls, R. G. (2008). Introduction to Simulation. Winter Simulation Conference.

[JRF12] Junges, R. and F. Klugl (2012). How to Design Agent-Based Simulation Models Using Agent Learning. Winter Simulation Conference.

[KEH09] Kehris, E. (2009). "Web-Based Simulation of Manufacturing Systems." International Journal of Simulation Modelling 8(2): 102-113.

[KIT12] kite (2012). "O que é MES?". Retrieved 11/02/2014, from http://www.kitemes.com.br/o-que-e-mes-manufacturing-execution-system/.

[KJP92] Kleijnen, J. P. C. (1992). "Verification and validation of simulation models." European Journal of Operational Research 82.

[KLJ07] Kletti, J. (2007). Manufacturing Execution System - MES.

[LMF10] Lizotte, M. and F. Rioux (2010). Image-Scenarization: A Computer-Aided approach for Agent-Based Analysis and Design. Winter Simulation Conference.

[MCP14] Manufacturing, C. (2014). "cmNavigo - MES." Retrieved 11/02/2014, from http://www.criticalmanufacturing.com/pt/products/.

[MCE14] Manufacturing, C. (2014). "Eficiência Operacional." Retrieved 11/02/2014, from http://www.criticalmanufacturing.com/pt/products/cmnavigo/Operational_efficiency/.

[MCI14] Manufacturing, C. (2014). "Integração Fabril." Retrieved 11/02/2014, from http://www.criticalmanufacturing.com/pt/products/cmnavigo/Factory_integration/.

[MCO14] Manufacturing, C. (2014). "Operations Intelligence." Retrieved 11/02/2014, from http://www.criticalmanufacturing.com/pt/products/cmnavigo/Operations_intelligence/.

[MCQ14] Manufacturing, C. (2014). "Qualidade Integrada." Retrieved 11/02/2014, from http://www.criticalmanufacturing.com/pt/products/cmnavigo/Integrated_quality/.

[MCR14] Manufacturing, C. (2014). "Rastreabilidade e Visibilidade em Tempo Real." Retrieved 11/02/2014, from http://www.criticalmanufacturing.com/pt/products/cmnavigo/Online_visibility/

[MAA97] Maria, A. (1997). Introduction to Modeling and Simulation. Winter Simulation Conference.

[MMG00] Mehrabi, M. G., et al. (2000). "Reconfigurable manufacturing systems: Key to future manufacturing." Journal of Intelligent Manufacturing (11): 403-419.

[MEH09] Meyer, H., et al. (2009). Manufacturing Execution Systems.

[NEM12] Neumann, M., et al. (2012). "Method for Multi-Scale Modeling and Simulation of Assembly Systems." Procedia CIRP 3: 406-411.

[NMW13] Neumann, M. and E. Westkämper (2013). "Method for Situation-based Modeling and Simulation of Assembly Systems." Procedia CIRP 7: 413-418.

[PLM04] Padgham, L. and M. Winikoff (2004). The Prometheus Methodology. Melbourne, Australia, RMIT University.

[PAP09] Pereira, A. P. A. (2009). Simulação de Sistemas de Produção Lean. Major de Automação, Faculdade de Engenharia da Universidade do Porto.

[RMJ13] Rabbani, M. J., et al. (2013). Modeling and Simulation Approach for an Industrial Manufacturing Execution System. 3rd International Conference on System Engineering and Technology. Shah Alam, Malaysia, IEEE.

[RAY08] Rao, Y., et al. (2008). On-Line Simulation for Shop Floor Control in Manufacturing Execution System. Hubei Province, China, University of Science and Technology.

[RFM11] Rioux, F. and M. Lizotte (2011). Image-Scenarization: From Conceptual Models to Executable Simulation. Winter Simulation Conference.

[RME12] Rolón, M. and E. Martínez (2012). "Agent learning in autonomic manufacturing execution systems for enterprise networking." Computers & Industrial Engineering 63(4): 901-925.

[RMM12] Rolón, M. and E. Martínez (2012). "Agent-based modeling and simulation of an autonomic manufacturing execution system." Computers in Industry 63(1): 53-78.

[RLT11] Rossetti, R.J.F., R. Liu and S. Tang (2011). Guest Editorial Special Issue on Artificial Transportation Systems and Simulation. IEEE Transactions on Intelligent Transportation Systems, 12 (2), pp. 309-312

[SUB09] Saenz de Ugarte, B., et al. (2009). "Manufacturing execution system – a literature review." Production Planning & Control 20(6): 525-539.

[SAP14] SAP (2014). "Dynamic versus Static Modeling Types." Retrieved 11/02/2014, from http://wiki.scn.sap.com/wiki/display/ModHandbook/Dynamic+versus+Static+Modeling+Types.

[SAM08] Savrasov, M. (2008). Overview of Flow Systems Investigation and Analysis Methods. The 8th International Conference “Reliability and Statistics in Transportation and Communication".

[SMA10] Services, M. A. M. R. (2010). MES: A Buyers Guide.

[SRE98] Shannon, R. E. (1998). Introduction to the Art and Science of Simulation. Winter Simulation Conference.

[SJM06] Simão, J. M., et al. (2006). "Manufacturing execution systems for customized production." Journal of Materials Processing Technology 179(1-3): 268-275.

[SRR02] Srivastava, R., et al. (2002). Stochastic vs. Deterministic Modeling of Intracellular Viral Kinetics, University of Wisconsin.

[SJD03] Sterman, J. D. (2003). Systems Dynamics: Thinking and Modeling for a Complex World. MIT Sloan School of Management, MIT Sloan School of Management.

[VPH00] Valckenaers, P. and H. Van Brussel Holonic Manufacturing Execution Systems. K. U. Leuven. Leuven, Belgium.

[WKT00] Worapradya, K. and T. Buranathiti "Integration of Manufacturing Execution System and Simulation."

Anexo A

Planeamento

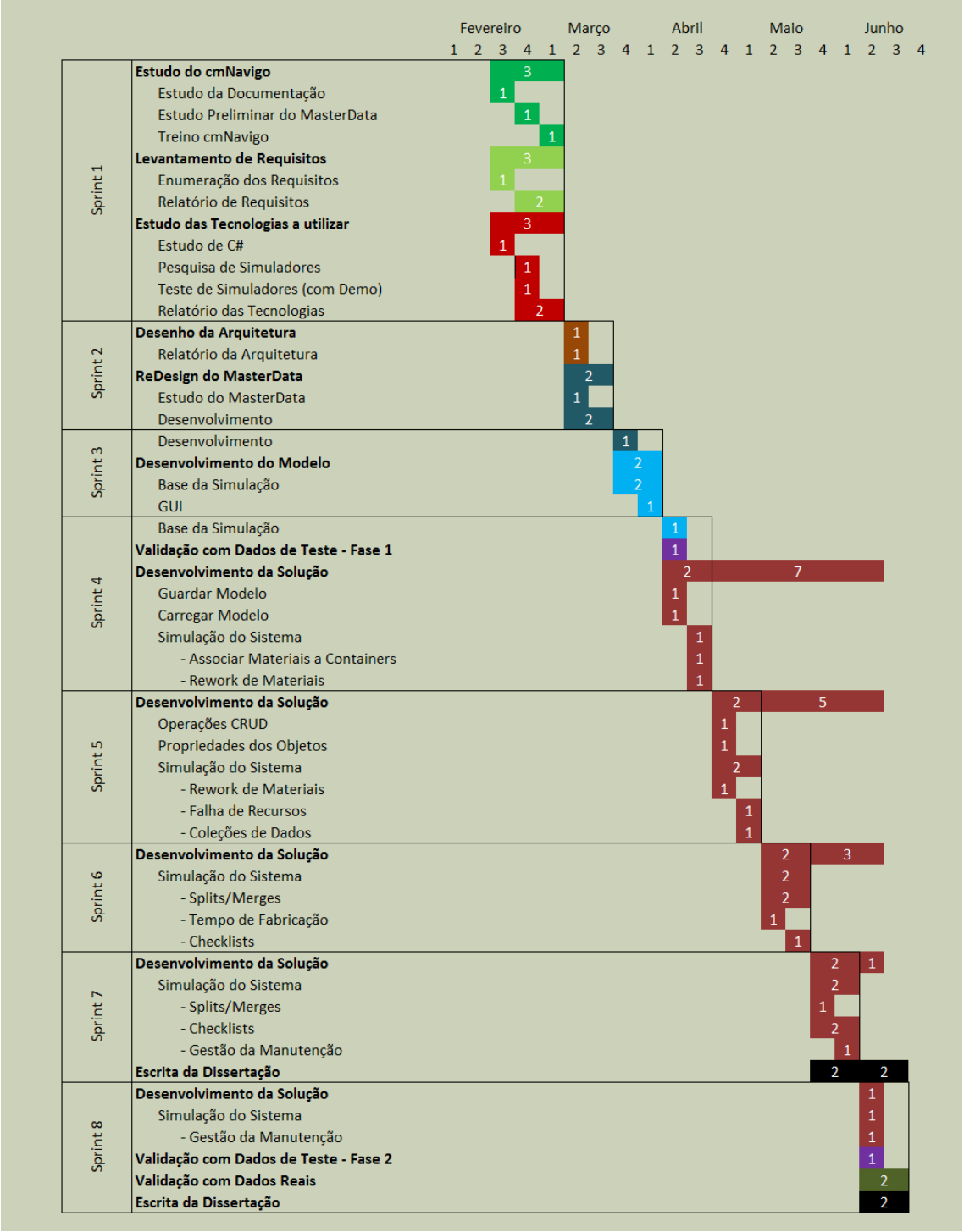


Figura 30: Planeamento do Trabalho

Anexo B

Principais Simuladores

Tabela 3: Principais simuladores de MES 1/6

Vendor Name	Product Names	Functional Range	Industries Served	Size of Business Served	Databases Supported	Platforms Supported	Delivery Mode
Apriso	FlexNet	Managing and executing production, warehouse, quality, maintenance, and labor activities	Automotive, aerospace and defense, clean-tech, consumer goods, industrial equipment, life sciences, packaging, electronics	Large	Microsoft SQL Server, Oracle	Has built-in Microsoft. NET with SOA; BPM	On-premises, virtual environment
Aspen Technologies	aspenONE	Includes data collection and storage, performance analysis, production dispatching, production resource management, production definition management, production execution	Chemicals, polymers, oil and gas, pharma, CPG, food and beverage	Small, midsize, and large	Microsoft SQL Server, Oracle	Microsoft Windows 2000, Windows XP, Windows Vista	On-premises
Camstar Systems	Camstar Enterprise Platform	Includes early collaborative manufacturing process development, capacity and capability analysis, integrated engineering change process, product/ WIP traceability, genealogy, audit trail, quality data collection and process limits, global change enforcement and audit trail	Medical devices, biotech, solar, semiconductor, electronics	Midsize and large	Microsoft SQL Server, Oracle	Microsoft	On-premises, SaaS
CDC Software	CDC Factory	Real-time finite production scheduling, real-time performance mgt., quality control, continuous improvement, business analytics, agile maintenance response, enterprise asset management, safety	Process mfg. and discrete mfg.	Midsize and large	Microsoft SQL Server, Oracle	Microsoft .NET, Windows 64	On-premises

Tabela 4: Principais simuladores de MES 2/6

Vendor Name	Product Names	Functional Range	Industries Served	Size of Business Served	Databases Supported	Platforms Supported	Delivery Mode
De Clercq Solutions	Objective MES	Dispatch work orders from ERP, production scheduling, manufacturing process mgt., non-conformance and corrective action, quality documentation, statistical analysis techniques, production reporting, supervisory control	Food and beverage, plastics, industrial parts and equipment, semiconductor, wholesale and distribution	Midsize and large	IBM DB/400, Microsoft SQL Server, Oracle, MySQL	HP/UX, IBM AIX, Sun Solaris/Sun OS, Linux, Microsoft Windows CE, Windows NT/2000/XP, Windows Server 2003, Windows Vista	On-premises
Emerson Process Management	Syncade Smart Operations Management Suite, AMS Suite (asset performance management)	Production operations, inventory mgt., quality mgt., maintenance mgt., integration	Chemicals, food and beverage, life sciences, metals and mining, oil and gas, and others	Small, midsize, and large	Microsoft SQL Server 2005, 2008	Microsoft Windows Server 2003 and 2008, Windows XP, Windows Mobile, 7, Windows Mobile, Palm OS, Android	On-premises
Eyelit	eyelit MES	Asset mgt., costing, shop floor control, reporting	Solar, semiconductor, MEMS, aerospace and defense, automotive, electronics	Midsize and large	Microsoft SQL Server, Oracle	Microsoft Windows, Unix, Linux, and others	Clients deploy using Java WebStart
GE Intelligent Platforms	Proficy Plant Applications	Includes order completion status, interactive schedule planning, automatic set-point loading, material delivery mgt.	Aerospace, automotive, chemicals, electronics, oil and gas, textile, constructions, CPG, and others	Small, midsize, and large	Microsoft SQL Server	Microsoft	On-premises

Tabela 5: Principais simuladores de MES 3/6

Vendor Name	Product Names	Functional Range	Industries Served	Size of Business Served	Databases Supported	Platforms Supported	Delivery Mode
HighJump Software	HighJump Manufacturing Advantage	Work order import, prioritization and dispatch; material picks and delivery to work cells; work queue; instruction and documentation review at work cells; material replenishments; real-time production review and adjustment, performance analysis; scrap reporting; machine utilization; labor productivity	Industrial mfg., automotive, aerospace, food and beverage	Midsize and large	Microsoft SQL Server, Oracle	Microsoft Windows	On-premises, cloud
Honeywell Process Solutions	Business FLEX, Matrikon, OptiVision, Uniformance	Includes planning and scheduling, supply chain mgt., operations mgt., data warehousing, integration and communications	Refining, oil and gas, power, chemicals, life sciences, and more	Small, midsize, and large	Microsoft SQL Server, Oracle	Any system from any vendor	On-premises
iBASEt	Solumina	Process planning, MES/MOM, quality management systems, supplier quality assurance, MRO	Aerospace, defense, nuclear products, shipbuilding, industrial equipment, industrial electronics, medical devices	Midsize and large	Microsoft SQL Server, Oracle	Microsoft Windows, Unix	On-premises
Intercim	Pertinence Suite powered by Velocity	Includes simplified process planning, advanced predictive analysis, process execution, quality mgt.	Aerospace, defense, discrete mfg., life sciences, high-tech, energy, and more	Small, midsize, and large	Microsoft SQL Server 2008 R2, Oracle 11g	Microsoft Windows Server 2008 (32/64-bit), Windows XP Professional, Windows Vista Business, Windows 7 (32-bit)	On-premises

Tabela 6: Principais simuladores de MES 4/6

Vendor Name	Product Names	Functional Range	Industries Served	Size of Business Served	Databases Supported	Platforms Supported	Delivery Mode
Invensys	Manufacturing Execution Module	Includes electronic work orders, BOM, product traceability, operations standard functionality, inventory control	Hybrid, process, discrete mfg.	Small, midsize, and large	Microsoft SQL Server 2005, 2008; Oracle	Microsoft SQL Server 2008, Windows Vista (64-bit)	On-premises
MPDV Mikrolab GmbH	HYDRA	Production data collection, shop floor scheduling, material and production logistics, machine data collection, tool management/DNC, quality assurance, process data collection, time and attendance, personnel scheduling, incentive pay, access control	Plastics, metal, automotive, food and beverage, plant and mechanical engineering, furniture, wood products, printing and packaging, precision mechanics and optics, electrical engineering and electronics	Midsize and large	Microsoft Access, Microsoft SQL Server, Oracle, MySQL, Informix, ASCII text file-based	HP/UX, Sun Solaris/ Sun OS, Linux, Microsoft NT/2000/XP, Windows Vista, Windows SQL Server 2003, Windows CE, Palm OS	On-premises
Parsec Automation	TrakSYS	Includes data/information integration management, process and asset/infrastructure modeling, production planning and dispatching, resource allocation, data acquisition, process and operation mgt.	Pharma, medical devices, food and beverage, consumer health products, automotive, chemicals, and others	Midsize and large	OLE-DB-compliant	Microsoft Windows, Linux, Unix	Browser technology/Web-based
Performix	Performix xRecipe, Performix xMES, Performix xBatch	Includes electronic work orders, multi-language support, OSI PI integration, material tracking, resource tracking, plant supervisory functions	Chemicals, consumer products, food and beverage, pharma	Midsize and large	Microsoft SQL Server	Microsoft Windows 2000, 2003, 2008, XP, Vista, and 7; Linux	On-premises

Tabela 7: Principais simuladores de MES 5/6

Vendor Name	Product Names	Functional Range	Industries Served	Size of Business Served	Databases Supported	Platforms Supported	Delivery Mode
Plex Systems	Plex Online	Includes production scheduling, finite scheduling, lean tools, shop floor control, bar-code labeling, traceability, labor/time tracking, SPCs	Automotive, medical devices, industrial mfg., food and beverage, life sciences, aerospace	Small and midsize	SaaS/cloud-based	SaaS/cloud-based	SaaS
Oracle	Oracle Manufacturing Execution System for Process Manufacturing, Oracle Manufacturing Execution System for Discrete Manufacturing	Execute, record, and monitor shop-floor activities in real time	Aerospace and defense, industrial mfg., high-tech, CPG, life sciences, oil and gas, and more	Midsize and large	Oracle Solaris x86-64 (64-bit)	Linux x86, Linux x86-64, HP-UX Itanium, HP-UX PA-RISC (64-bit), IBM AIX on Power Systems (64-bit), Microsoft Windows Server (32-bit), Oracle Solaris SPARC (64-bit)	On-premises, on-demand, or hybrid
Rockwell Automation	FactoryTalk ProductionCentre	Scheduling, order mgt., quality control and mgt., material tracking and mgt., WIP/Inventory, workflow and performance mgt.	Pharma, biotech, medical devices, CPG, food and beverage, automotive	Midsize and large	Oracle 10, 11; Microsoft SQL Server 2005, 2008; Windows 2003, 2008, including 64-bit; Linux; Solaris Enterprise	Oracle 10, 11; Microsoft SQL Server 2005, 2008; Windows 2003, 2008, including 64-bit; Linux; Solaris Enterprise	On-premises, SaaS

Tabela 8: Principais simuladores de MES 6/6

Vendor Name	Product Names	Functional Range	Industries Served	Size of Business Served	Databases Supported	Platforms Supported	Delivery Mode
SAP AG	SAP Manufacturing Execution	Traceability, non-conformance mgt., production transfer, ERP integration, return and repair, labor tracking, engineering change mgt., production metrics, globalization	High-tech, industrial machinery and components, aerospace and defense, automotive, medical devices	Small, midsize, and large	Microsoft SQL Server, Oracle	Microsoft Windows Server, HP-UX, Solaris, Linux, AIX	On-premises
Schneider Electric	Ampla	Metrics, production, downtime, quality, inventory, energy, planning, cost, knowledge, maintenance	Mining, food and beverage, CPG, water/wastewater	Midsize and large	Microsoft SQL Server, Oracle, OPC DA, OPC HDA, InSQL, major SCADA and historian systems	Microsoft, Java	CD
Siemens MES	SIMATIC IT	Includes quality control, traceability, process efficiency, production mgt., collaboration	Electronics, automotive, assembly, food and beverage, and others	Midsize and large	Microsoft SQL Server, Oracle	Microsoft Windows XP, Windows 7	On-premises, Web-based
Werum Software & Systems	PAS-X	Master batch records, finite scheduling, weighing and dispensing, electronic batch recording, equipment mgt., material track-and-trace, warehouse mgt., process quality control, corrective and preventive actions, operator training records, manufacturing intelligence	Pharma, biotech	Midsize and large	Oracle	Flexible	On-premises

Anexo C

Review Planning

1. Question Formularization

1.1 Question Focus:

The focus of this Systematic Review will be to gather information on the different existing MES, how to tailor a solution to cmNavigo and in the end to ascertain whether it is better to adopt an existing solution or to create a new one.

1.2 Question Quality and Amplitude:

Problem

Lack of an appropriate MES that fully integrate with a solution already developed at the company, namely cmNavigo, that is lighter and easier to use and can bring competitive advantages to its owner; Lack of means to automate configuration set-up of clients' implemented solutions to assess current systems and suggest improvements.

Questions

1. What are the simulators of Manufacturing Execution System (MES) currently available and mostly used by similar industries?
2. If indeed there exist available simulators, are they able to fulfill CM's requirements for this sort of application?
3. If there aren't any available simulators, how to devise and implement one that it's lighter, simpler and easier than the ones currently known?

Keywords and acronyms

- Manufacturing Execution Systems; MES,
- Industrial Simulator,
- industrial process,
- competitive advantages,
- production flows,
- backwards simulation,
- machines,
- stress test,

- demonstration systems,
- test systems,
- proof of concept,
- system operation,
- manufacturing routes,
- environments and simulation,
- production management,
- simulation of the manufacturing operation

Especially (Solution will certainly include the following ideas):

- Visual Interactive Modeling (VIM)
- Visual Interactive Simulation (VIS)
- Object-oriented modeling and simulation
- Model-driven and Data-driven simulation of industrial systems
- Intelligent Simulation (for industrial applications)
- Test, Verification, Calibration and Validation of Simulation Models (for industrial applications)

2. Sources Selection

2.1 Sources Selection Criteria Definition

- Prominent Scientific Journals and Conference Proceedings in related areas (Journals such as: SIMULATION – Transactions of SCS; Conferences such as: Winter Simulation Symposium; Spring Simulation Conference; SIMULTECH; I3MM; Conferences by SCS, SCS-Europe, EROSIS, IEEE, ACM; ESM; EMSS; ISC - The Industrial Simulation Conference)
- Papers must be available for download for free (both open-access and through FEUP's b-On Portal)
- Papers must propose or discuss a strategy about MES
- Papers must focus on industrial simulation

2.2 Sources Identification

- Search Methods
- Search through web search engines
- Search Strings

Also include Web sites of companies that develop such systems: Arena; Simul8; GPSS; Simulink; among others; If possible, look for their "manuals; white papers; demo licenses; etc." If possible, test with some of such applications to see whether they offer the required features.

The Search Strings for these questions are:

(1) - ((“MES” OR “Manufacturing Execution System”) AND ((“MES” OR “Manufacturing Execution System”) AND (“industrial” OR “production” OR “factory” OR “industrial process” OR “industrial simulator”))

(2) - ((“MES” OR “Manufacturing Execution System”) AND ((“MES” OR “Manufacturing Execution System”) AND (“industrial” OR “production” OR “factory” OR “industrial process” OR “industrial simulator” OR “machines” OR “demonstration systems” OR “manufacturing routes” OR “production management” OR “environments and simulation” OR “simulation of the manufacturing operation”))

(3) - ((“MES” OR “Manufacturing Execution System”) AND ((“MES” OR “Manufacturing Execution System”) AND (“industrial” OR “production” OR “factory” OR “industrial process” OR “industrial simulator” OR “machines” OR “demonstration systems” OR “manufacturing routes” OR “production management” OR “environments and simulation” OR “simulation of the manufacturing operation”)) AND ((“MES” OR “Manufacturing Execution System”) AND (“industrial” OR “production” OR “factory” OR “industrial process” OR “industrial simulator” OR “machines” OR “demonstration systems” OR “manufacturing routes” OR “production management” OR “environments and simulation” OR “simulation of the manufacturing operation”) AND (“lighter” OR “simpler” OR “easier” OR “modern” OR “current”)))

Anexo D

Relatório de Requisitos

REQUIREMENTS REPORT

MES Simulator

Project Code:	MES-S
Doc. Reference:	<CMF-PPPP-2011-TTT-NNNN>
Version:	1.00
Author:	João Pinto, joao.pinto@criticalmanufacturing.com

Status:	Draft, Date: 2014-02-25
Approval:	João Brandão, ja-brandao@criticalmanufacturing.com, Principal Software Engineer

Access:	Internal Use Only Internal Access: Project Team, Quality, Management External Access: <TBD>

1 Contents

2 Introduction 3

3 Simulator Requirements 5

2 Document Version

Number	Date	Changes
1	24/02/2014	Creation of the document with the early requirements.
2	24/03/2014	Changes to the following requirements: Duplicate Objects – Removed Object Properties – Removed Reset Simulation – Removed RF13 - RF20 – Added

3 Introduction

The range of existing simulators already in the market is quite wide, going from simple simulators that nevertheless fulfil their job to simulators in three dimensions where the ability to customize the model is so big that resemble games in the genre. Therefore is essential to carry out a study in which the goal is to identify the requirements of the project in order to be possible to restrict the choices (simulators) found. Accordingly, this report, where it's presented a list of the main functional requirements, was elaborated.

4 Simulator Requirements

The requirements presented in this section are structured as described in Table 1.

Title	Description
ID	The requirement unique identifier
Title	The title of the requirement
Definition	A brief explanation of the requirement
MoSCoW	MoSCoW evaluation of the requirement

Table 1: Requirements Structure

ID	Title	Definition	MoSCoW
RF01	Model of Production Line	The client must be able to model his production line in the simulator through the loading of the MasterData.	Must
RF02	Load MasterData	The application must load data located in an excel file, called MasterData, into the simulator.	Must
RF03	Save MasterData	The application must save the data at the end of the simulation into an excel file, creating a MasterData.	Must
RF04	Save Model	The application should permit the user to save the actual model so that he can use it in another environment.	Should
RF05	Load Model	The application should let the user load an existing model into the simulator.	Should
RF06	WCF Communication	The application must allow WCF communication so that cmNavigo can be integrated.	Must
RF07	Simulate System	The client must be able to simulate the operations that cmNavigo can do in his production line.	Must
RF08	Data Analysis	The application should allow the client to analyse some data extracted from the simulation, either in real-time or at the end.	Should

RF09	Production Line Visualization	The application must permit a visualization of the clients' production line through the use of fabLive	Should
RF10	Stress Tests	The simulation should be able to perform stress tests to the client's current and modified production line model.	Should
RF11	CRUD Operations	The application should allow the user to perform CRUD operations in the objects of the production line model.	Should
RF12	Real-Time Visualization	The application could permit a real-time visualization of the simulation while it's running.	Could
RF13	Associate Material Container	It should be possible to associate a material to a container, choosing if the material will stick with the container through the production line or not.	Should
RF14	Material Rework	It should be possible choose which material will have to do a rework and in which step it will do it.	Should
RF15	Resource Failure	It should be possible to choose one or more resources to fail, by track in count or by time.	Should
RF16	Data Collections	The Data Collections for track in, track out, rework and move next should be processed.	Should
RF17	Split/Merge	It should be possible to choose a probability for a material to split into two, as long as it has the quantity to do it. The reverse (merge) should also be possible.	Should
RF18	TrackIn Yield and Cycle Time	The track in yield and cycle time should be respected.	Should
RF19	Checklists	The Checklists for track in, track out, rework and move next should be processed.	Should
RF20	Maintenance Management	A resource that is down should be able to return to a standby state by performing a Maintenance Plan.	Should
RF21	Performance Advices	The application could suggest the user some improvements to his current production line, based on the simulation just performed.	Would

Table 2: Requirements of the Project

Anexo E

Relatório de Arquitetura

ARCHITECTURE REPORT

MES Simulator

Project Code:	MES-S
Doc. Reference:	<CMF-PPPP-2011-TTT-NNNN>
Version:	1.00
Author:	João Pinto, joao.pinto@criticalmanufacturing.com

Status:	Draft, Date: 2014-03-06
Approval:	João Brandão, ja-brandao@criticalmanufacturing.com, Principal Software Engineer

Access:	Restricted Internal Access: Project Team, Quality, Management External Access: TBD

1 Contents

2 Introduction 3
2.1 Version History 3
2.2 Goal and Scope 3
2.3 Structure 3
3 Architecture 4
3.1 Logical Architecture 4
3.1.1 Relational Model 4
3.1.2 Modules 7
3.2 Physical Architecture 8
4 Technologies 9
4.1 Windows presentation Foundation 9
4.2 C♯ 9
4.3 WCF 9
4.4 .Net Framework 9

2 Introduction

2.1 Version History

Version	Date	Description
1.00	06/03/2014	First version with sketches of the physical and logical architecture, as well as a brief description of the technologies used.

2.2 Goal and Scope

After the elicitation of system requirements is essential to organize and design the system architecture. A well-defined architecture becomes essential and will aid the implementation phase.

This report aims to provide a better understanding of some technical aspects of the project, such as how the system is structured and divided. It also will help understand what and how the technologies discussed are interconnected, their characteristics and features, thus obtaining an overview of the system architecture.

This project comes as part of the dissertation of the MSc in Informatics and Computing Engineering at FEUP, being developed for Critical Manufacturing. One of the main products developed by CMF is cmNavigo. This software is a Manufacturing Execution System entirely based on Microsoft technologies that allows the user to know whether what was planned by the ERP system is being properly implemented and if not why is that happening, where the problems, and even the type of problems that occur most frequently, are occurring. Being this software in direct competition with some of the top products in the world of its kind, and given the youth of the solution necessarily with a customer base of lower reference, it is necessary to introduce this innovative and differentiating feature.

The core of this project consists of the design and development of a solution of configuration and simulation of cmNavigo for testing, demonstration for clients and proofs of concept, with the end result leading to a solution that will bring great competitive advantages for the company.

2.3 Structure

This report is divided into chapters which are described below. The first chapter is the introduction of the document. The second one is the description of the system architecture. The first one is the logical architecture, where will be explained the interaction between all modules and structural organization. Then the physical architecture, where systems will be identified using the project as well as a high-level view of the physical components will be presented. The third and last chapter will introduce the development technologies, including the reasons for their choice and how they bring value to the project.

3 Architecture

3.1 Logical Architecture

In this chapter is presented the high level view of the system architecture to be developed and is divided into two sections: in the first one the relational model of the main objects of the systems is presented, along with all the objects that are created directly from the extraction of the MasterData file; and the second one shows the three main models of the system and how their main tasks.

3.1.1 Relational Model

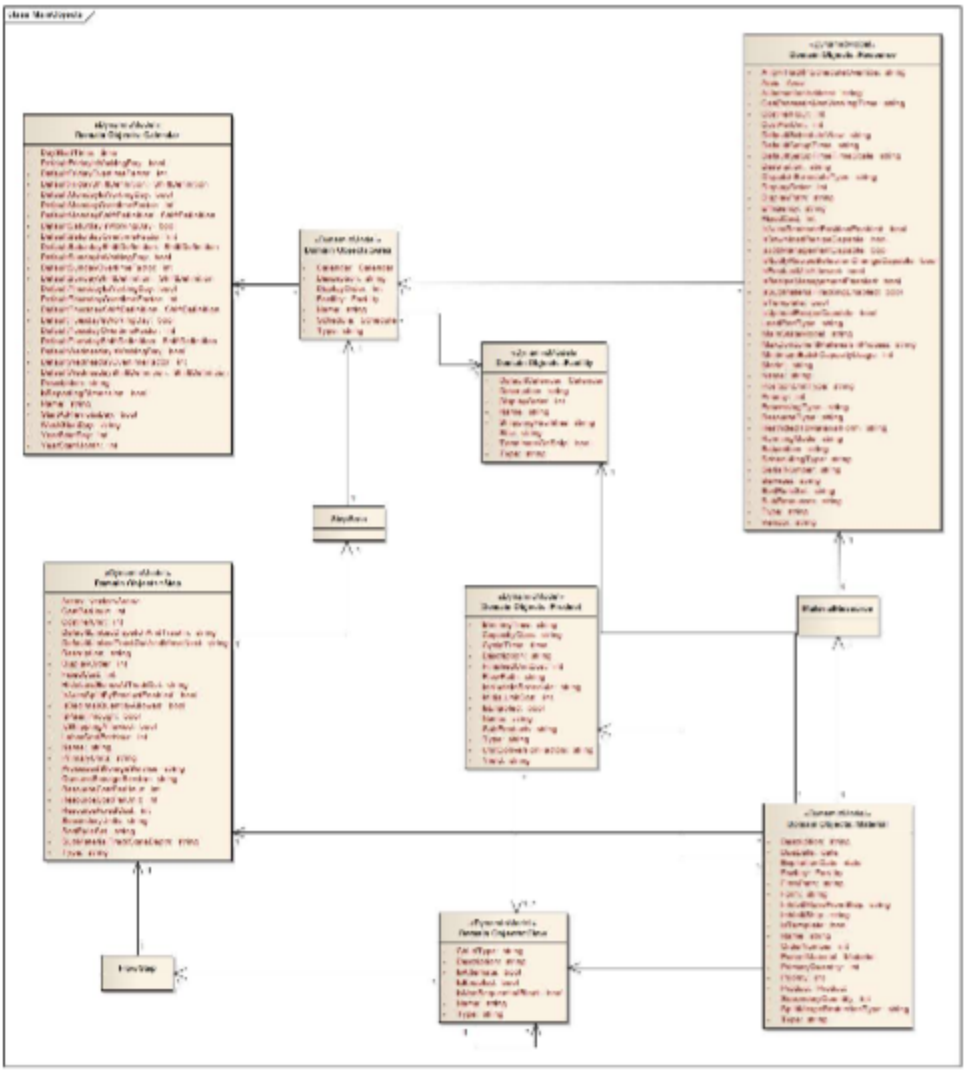


Figure 1: Main objects of the system and their relations

Figure 1 shows the relational model of the main objects which the system is built upon. The Material is the center of the factory. It's a raw material that after several Steps is transformed into a Product. These Steps are arranged into Flows, so a Material is not directly connected with a Step, but instead is connected with the Flow that contains those Steps. A Resource is an entity that participates in the transformation of a Material into a Product, so the Material is also connected with a specific Resource at a given time. It's also important to note that a Material can be at the same time a group of Materials. Finally, a Material is processed at a specific Facility, which in turn contains one or more Areas that have the Calendar that controls the production.

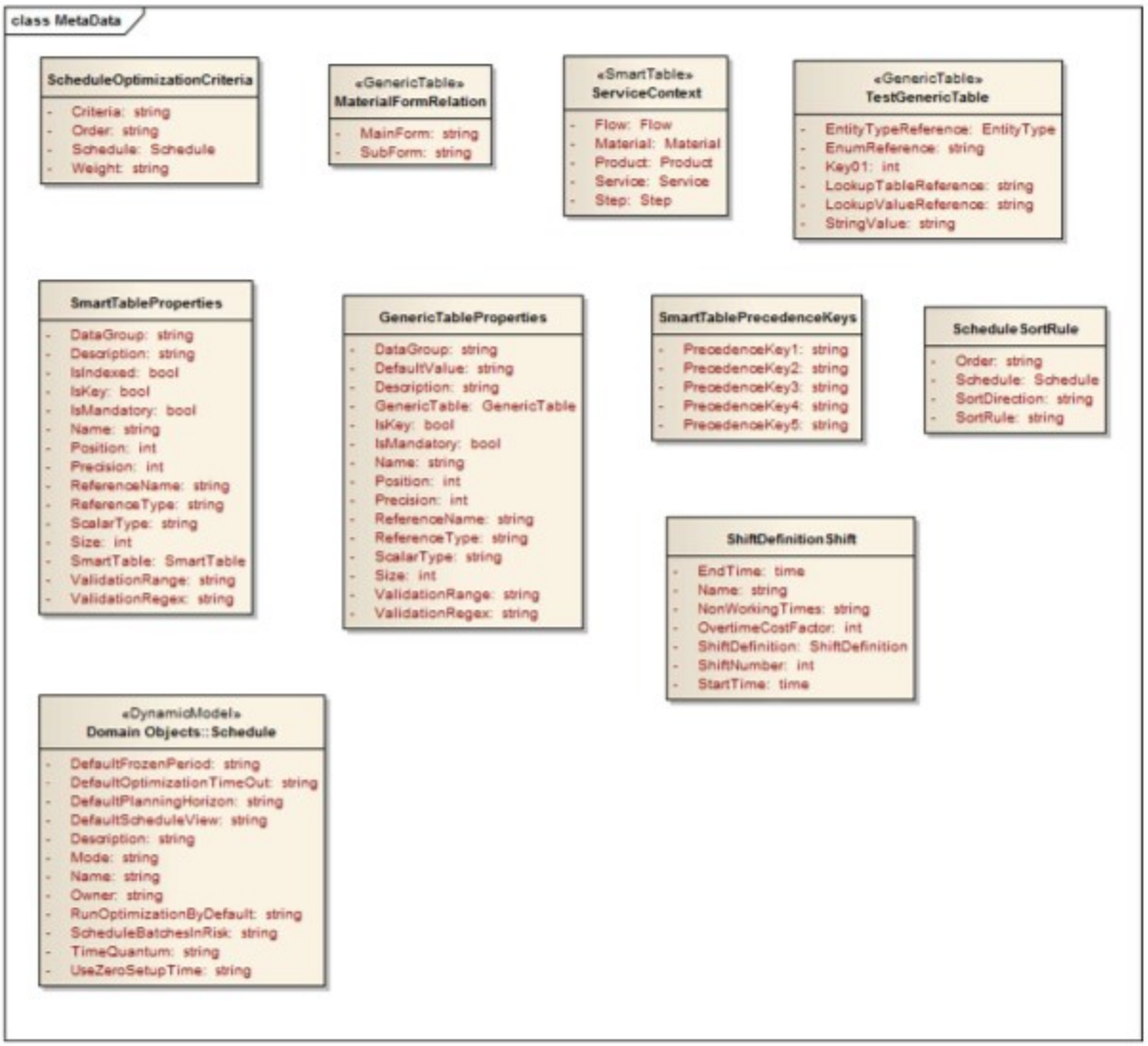


Figure 2: Objects that include metadata and doesn't have a specific order to be loaded.

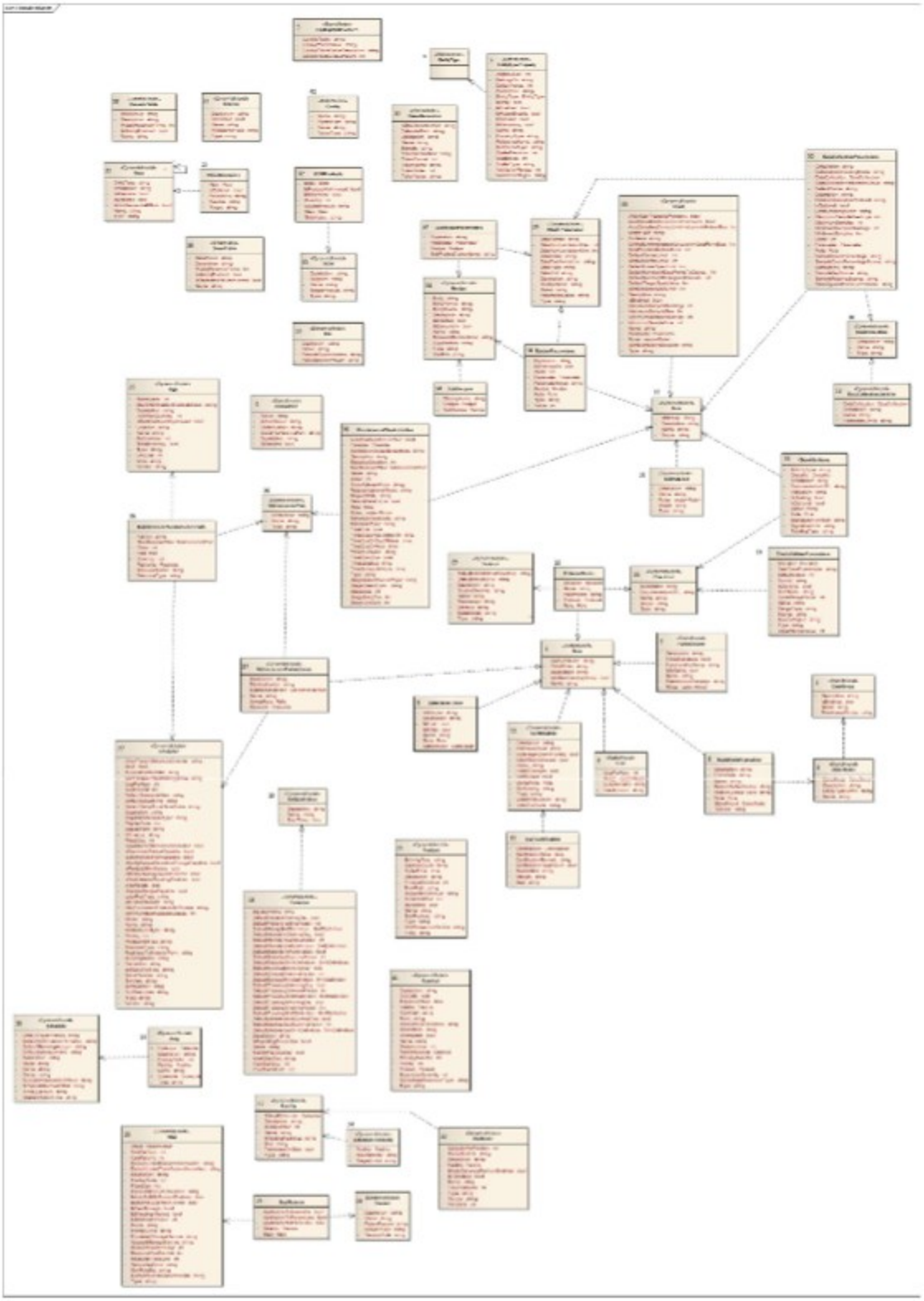
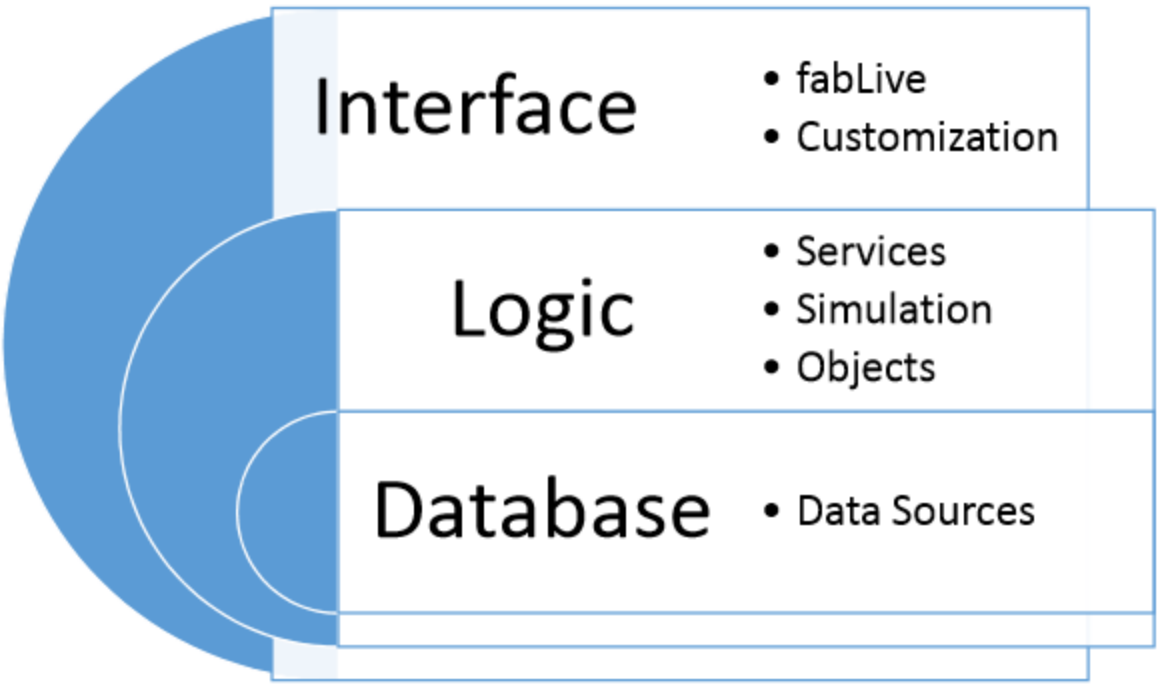


Figure 3: All objects that are possible to load from MasterData

Figure 2 and 3 shows all of the objects that are possible to load from MasterData and their relations. There are more objects that are created at the same time the MasterData is processed but they aren't in these schemas.

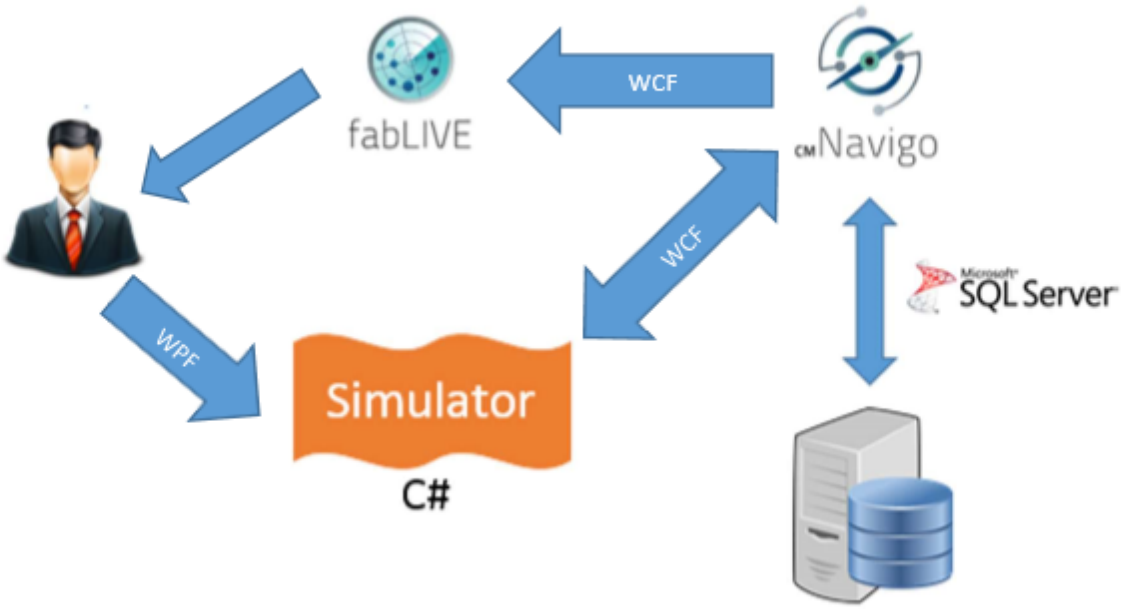
3.1.2 Modules

It is simpler to understand the overview of the system if it's divided into big sections of functionalities. The logic of the system is divided in three big groups, as shown in Figure 4. The Interface that contains all elements that are visible to the user, including the visualization in real time of the simulation through fabLive and the customization that will be possible to do; the logic that rules it all, including the services that communicate with cmNavigo via WCF, the logic of the simulation itself and the objects that are already discussed above; and the Database which have all the information of the simulation and is updated also in real time.



3.2 Physical Architecture

Regarding the physical architecture, it's possible to see three very different stages of the process. The first one is the user interaction with the system. He will be able to customize de simulation beyond a simple execution of the production line. It'll also be possible to check the simulation in real-time, using fabLive to do that. The second stage is the simulator itself. It'll do the communication between the interface and cmNavigo, besides the simulation. The last one is cmNavigo and CMF servers, where the information will be stored.



4 Technologies

Critical Manufacturing only works with Microsoft technologies, so the choice of the ones being used was already very restricted. Analysing the previous architecture, we came to the conclusion that the tool will have the three stages already mentioned. The interface will be built using Windows Presentation Foundation taking advantage of the fact that it's the easier way to do a GUI, without needing to be a web application. The programming language chosen for the simulation (and for Silverlight) is C#, same one as cmNavigo and fabLive. The communication between the simulator and cmNavigo services will have to be built using Windows Communication Foundation (WCF). Finally, database services (if needed) will use Microsoft SQLServer.

4.1 Windows presentation Foundation

Windows Presentation Foundation (or WPF) is a graphical subsystem for rendering user interfaces in Windows-based applications by Microsoft. WPF, previously known as "Avalon", was initially released as part of .NET Framework 3.0. Rather than relying on the older GDI subsystem, WPF uses DirectX. WPF attempts to provide a consistent programming model for building applications and separates the user interface from business logic. It resembles similar XML-oriented object models, such as those implemented in XUL and SVG.

4.2 C♯

C♯ is a multi-paradigm programming language encompassing strong typing, imperative, declarative, functional, procedural, generic, object-oriented (class-based), and component-oriented programming disciplines. It was developed by Microsoft within its .NET initiative and later approved as a standard by Ecma (ECMA-334) and ISO (ISO/IEC 23270:2006). C♯ is one of the programming languages designed for the Common Language Infrastructure and is built on the syntax and semantics of C++, allowing C programmers to take advantage of .NET and the common language runtime.

4.3 WCF

WCF is a tool often used to implement and deploy a service-oriented architecture (SOA). It is designed using service-oriented architecture principles to support distributed computing where services have remote consumers. Clients can consume multiple services; services can be consumed by multiple clients. Services are loosely coupled to each other. Services typically have a WSDL interface (Web Services Description Language) that any WCF client can use to consume the service, regardless of which platform the service is hosted on. WCF implements many advanced Web services (WS) standards such as WS-Addressing, WS-ReliableMessaging and WS-Security. With the release of .NET Framework 4.0, WCF also provides RSS Syndication Services, WS-Discovery, routing and better support for REST services.

4.4 .Net Framework

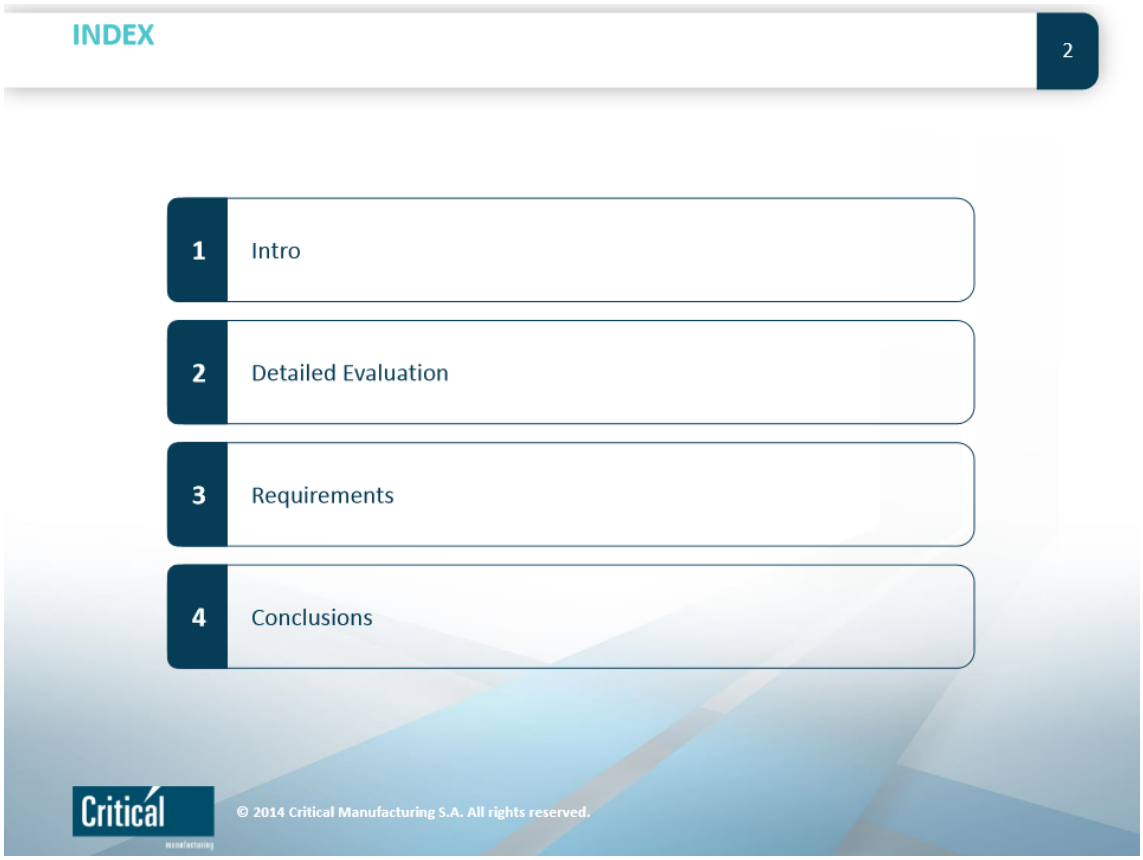
Microsoft SQL Server is a relational database management system developed by Microsoft. As a database, it is a software product whose primary function is to store and retrieve data as requested by other software applications, be it those on the same computer or those running on another computer across a

network (including the Internet). There are at least a dozen different editions of Microsoft SQL Server aimed at different audiences and for workloads ranging from small single-machine applications to large Internet-facing applications with many concurrent users. Its primary query languages are T-SQL and ANSI SQL.

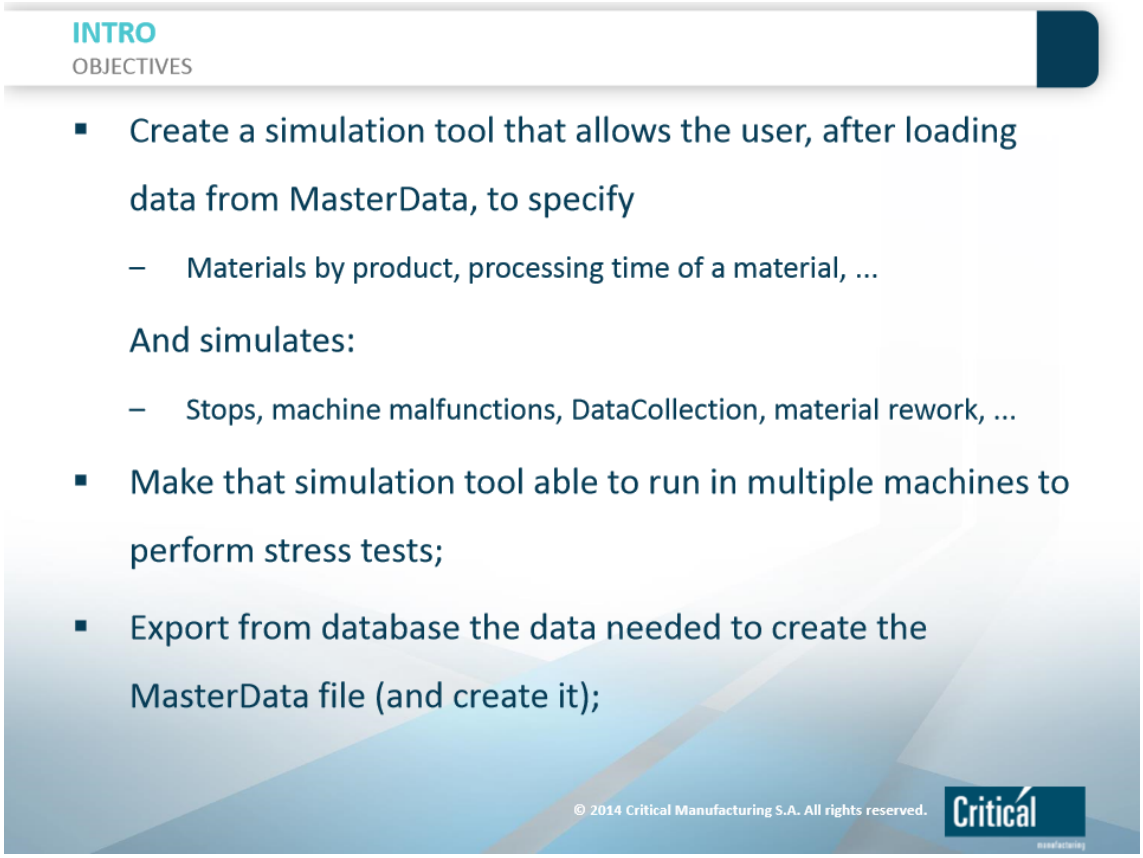
Anexo F

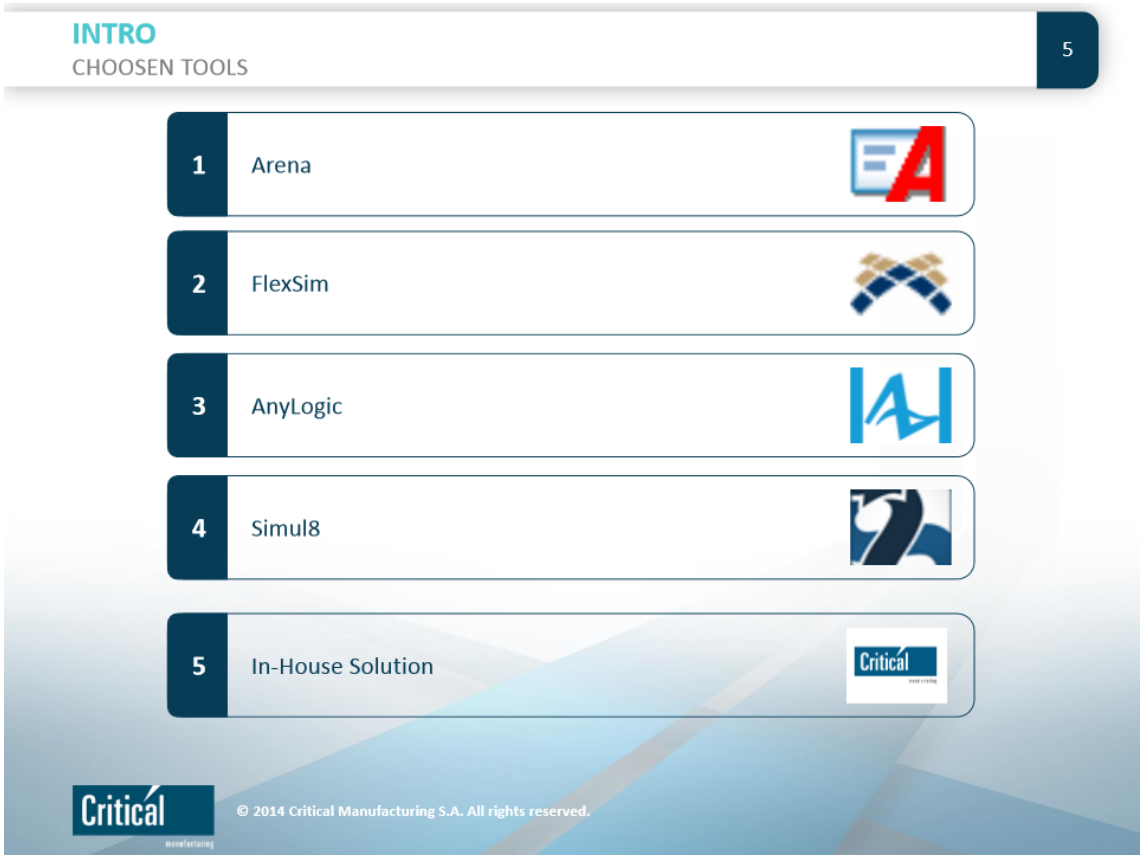
Relatório de Tecnologias


Critical
manufacturing
unleash the
power of
manufacturing
CRITICAL
MANUFACTURING
SIMULATORS
COMPARISON
© 2014 Critical Manufacturing S.A. All rights reserved.


INDEX
2
1 Intro
2 Detailed Evaluation
3 Requirements
4 Conclusions
Critical
manufacturing
© 2014 Critical Manufacturing S.A. All rights reserved.


Critical
manufacturing
3
unleash the power of manufacturing
SIMULATORS COMPARISON
INTRO
© 2014 Critical Manufacturing S.A. All rights reserved.


INTRO
OBJECTIVES
Create a simulation tool that allows the user, after loading data from MasterData, to specify
– Materials by product, processing time of a material, ...
And simulates:
– Stops, machine malfunctions, DataCollection, material rework, ...
Make that simulation tool able to run in multiple machines to perform stress tests;
Export from database the data needed to create the MasterData file (and create it);
© 2014 Critical Manufacturing S.A. All rights reserved.
Critical
manufacturing


INTRO
CHOOSEN TOOLS
5
1 Arena
2 FlexSim
3 AnyLogic
4 Simul8
5 In-House Solution
Critical
manufacturing
© 2014 Critical Manufacturing S.A. All rights reserved.


Critical
manufacturing
6
unleash the power of manufacturing
SIMULATORS COMPARISON
DETAILED EVALUATION
© 2013 Critical Manufacturing S.A. All rights reserved.

DETAILED EVALUATION
SOURCES

7

Trials/Demos
Experimenting the software, doing some examples and demos.

Direct Contact
Direct contact with the vendors in order to get informations they don't provide in the website.

Information from Site
Each product website has some relevant information about the software.

Web Search
More but discarded solutions were found through web seacrh.

Papers and Journals
Consulting some papers and journals in the field was one of the best methods to gather information.

Guides
To learn how to proper work with the software, the application guides and some videos were studied.

DETAILED EVALUATION
IN-HOUSE: PROS AND CONS

8

Advantages	Limitations
Allows WCF communication	Solution that doesn't have market proofs
Can be adapted to meet all Critical needs and requirements	High risk of developing software from scratch
Solution adapted to the Critical coding guidelines	Can become obsolete if not upgraded regularly
Can be adapted to each new product developed by Critical	Will not have the level of functionality compared to a commercial-off-the-shelf product

DETAILED EVALUATION
TASKS' PRICE

9

ID	Name	Description	Risk	Cost (in man/hours)
T01	Technologies' Study	An initial study of the technologies to be used has to be made. This includes the study of cmNavigo, the programming languages to be used, the Stress Test and MasterData already developed and the different simulators in the market.	High	64 - 8 days
T02	Architecture Design	Since this is a one man job, the arquitechure will serve more to explain to the client what's the global vision of the solution, nevertheless this is an essential step of every project.	Low	32 - 4 days
T03	Technical Requirements	The requirements already agreed are only macro-tasks, the technical ones are yet to be discussed.	Medium	16 - 2 days
T04	Model Development	In order to achieve the best simulation possible, it's necessary to have a robust simulation model behind. So is devoted enough time to choose the best approach, as well as the design of the model for that approach.	High	80 - 10 days

Critical manufacturing

DETAILED EVALUATION
TASKS' PRICE

10

ID	Name	Description	Risk	Cost (in man/hours)
T05	Solution Development	Design and development of the simulator and respective integration with cmNavigo. This macro-task will be divided as specified on T03.	High	360 - 45 days
T06	Tests with Test Data	It is important to validate the application with test data to be able to understand if each module is operating correctly. There will be two stages of validation with test data, the first one to validate the model and the second to validate the simulator.	Medium	40 - 5 days
T07	Tests with Real Data	The validation will only be completed when it is possible to simulate a real factory with precision. In order to do that it's necessary to test the solution with data from a real company.	High	80 - 10 days
T08	Documentation	The final documentation is the final step of every project. All of the steps that lead to the development of the solution have to be documented and an user-manual has to be written.	Low	64 - 8 days

Critical manufacturing

REQUIREMTNS
DETAILS

ID	Title	Definition	MoSCoW
RF01	Model of Production Line	The client must be able to model his production line in the simulator through the loading of the MasterData.	Must
RF02	Load MasterData	The application must load data located in an excel file, called MasterData, into the simulator.	Must
RF03	Save MasterData	The application must save the data at the end of the simulation into an excel file, creating a MasterData.	Must
RF04	Save Model	The application should permit the user to save the actual model so that he can use it in another environment.	Should
RF05	Load Model	The application should let the user load an existing model into the simulator.	Should
RF06	WCF Communication	The application must allow WCF communication so that cmNavigo can be integrated.	Must
RF07	Simulate System	The client must be able to simulate the operations that cmNavigo can do in his production line.	Must
RF08	Data Analysis	The application should allow the client to analyse some data extracted from the simulation, either in real-time or at the end.	Should
RF09	Production Line Visualization	The application must permit a visualization of the clients' production line through the use of fabLive	Should
RF10	Stress Tests	The simulation should be able to perform stress tests to the client's current and modified production line model.	Should
RF11	CRUD Operations	The application should allow the user to perform CRUD operations in the objects of the production line model.	Should
RF12	Real-Time Visualization	The application could permit a real-time visualization of the simulation while it's running.	Could
RF13	Performance Advices	The application could suggest the user some improvements to his current production line, based on the simulation just performed.	Would

Critical manufacturing

REQUIREMENTS
COMPARISON

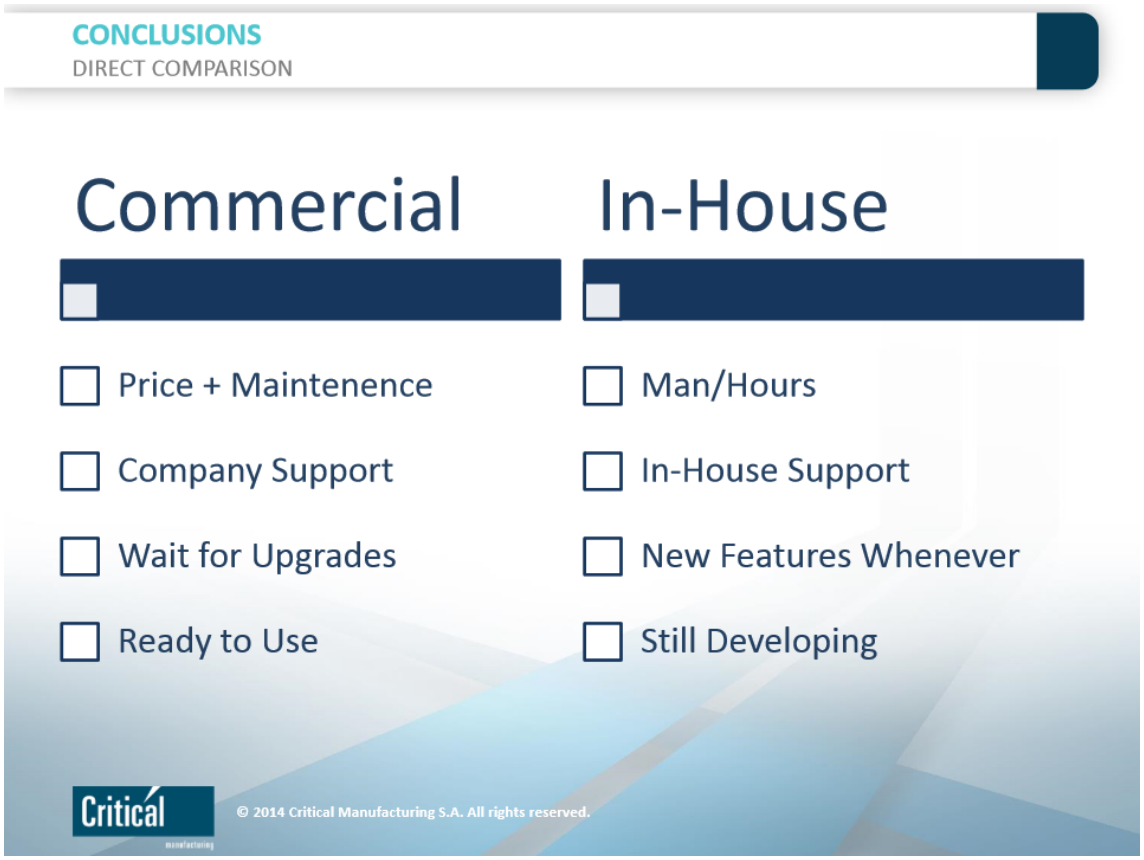
	Arena	FlexSim	AnyLogic	Simul8	In-House
RF01	✓	✓	✓	✓	✓
RF02	✓	✓	✓	✓	✓
RF03	✗	✗	✗	✗	✓
RF04	✓	✓	✓	✓	✓
RF05	✓	✓	✓	✓	✓
RF06	✗	✗	✗	✗	✓
RF07	✓	✓	✓	✓	✓
RF08	✓	✓	✓	✓	✓
RF09	✗	✗	✗	✗	✓
RF10	✓	✓	✓	✓	✓
RF11	✓	✓	✓	✓	✓
RF12	✓	✓	✓	✓	✓
RF13	✗	✗	✗	✗	✓

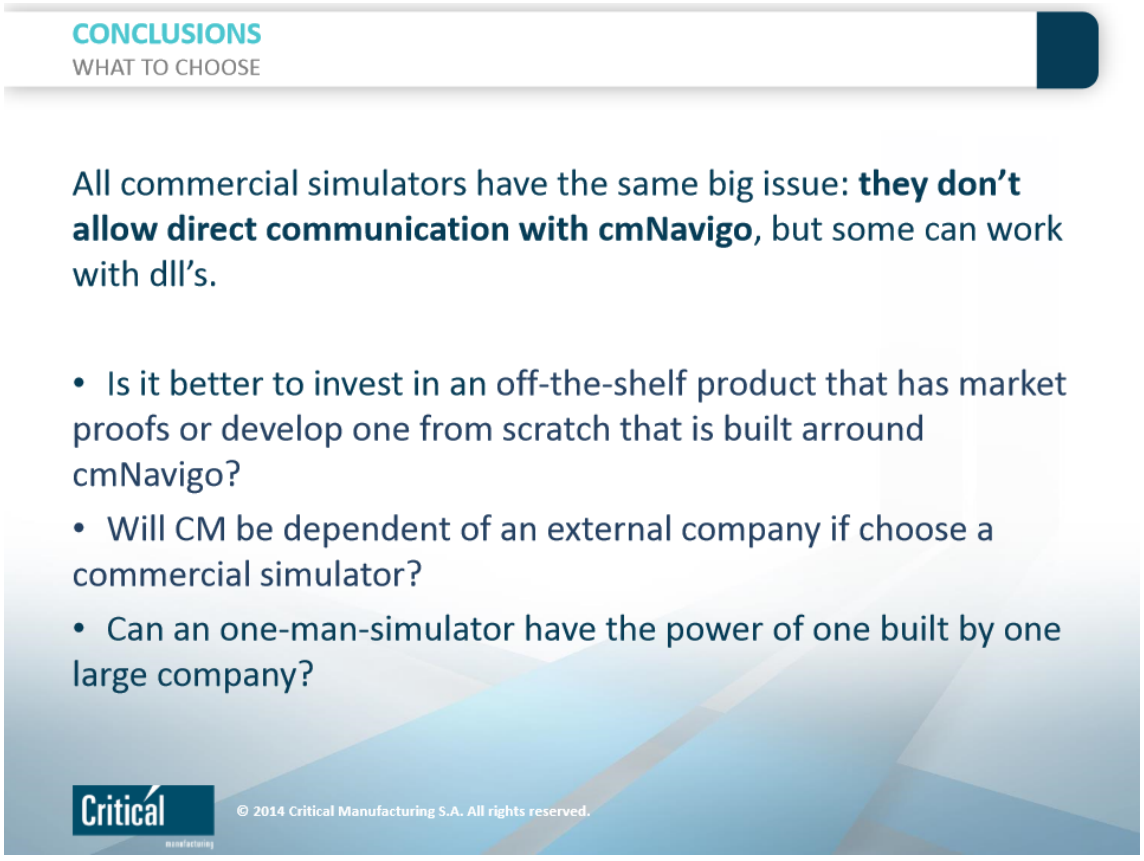
Critical manufacturing

Critical manufacturing

14

SIMULATORS COMPARISON CONCLUSIONS


CONCLUSIONS
DIRECT COMPARISON
Commercial
In-House
Price + Maintenence
Company Support
Wait for Upgrades
Ready to Use
Man/Hours
In-House Support
New Features Whenever
Still Developing
Critical
© 2014 Critical Manufacturing S.A. All rights reserved.


CONCLUSIONS
WHAT TO CHOOSE
All commercial simulators have the same big issue: they don't allow direct communication with cmNavigo, but some can work with dll's.
• Is it better to invest in an off-the-shelf product that has market proofs or develop one from scratch that is built arround cmNavigo?
• Will CM be dependent of an external company if choose a commercial simulator?
• Can an one-man-simulator have the power of one built by one large company?
Critical
© 2014 Critical Manufacturing S.A. All rights reserved.

Anexo G

Objetos do cmNavigo

Tabela 9: Objetos que pertencem ao Master Loader

Nome	Tipo	Descrição
Role	*Static Model*	É um objeto que representa o papel de um utilizador, por exemplo um Admninistrador
Functionality	*Static Model*	Funcionalidades adicionais do sistema.
DataGroup	*Static Model*	Grupos de *Roles* que podem aceder a certos dados.
User	*Static Model*	Utilizadores inscritos no sistema.
LookupTableValues	*Static Model*	Valores de uma das tabelas que pertencem ao sistema.
EntityType	*Static Model*	Uma *EntityType* representa um objeto do sistema e contém toda a meta-data que lhe está atribuída.
EntityTypeProperty	*Static Model*	As propriedades de cada *EntityType*.
StateModel	*Static Model*	Um modelo com diferentes estados
StateModelStates	-	Estados de cada *StateModel*
StateModelTransitions	-	Transições entre cada *StateModel*
DEEAction	*Static Model*	Uma *Dynamic Execution Engine Action* são regras escritas em C# que são carregadas e compiladas para o cmNavigo para serem executadas em runtime.
NameGenerator	*Static Model*	É um mecanismo de gerar nomes aleatórios. Contém propriedades que podem ser definidas para personalizar os nomes a gerar.
Certification	*DynamicModel*	Representa uma determinada habilidade, conhecimento ou treino que é

		atribuído a um *User*.
UserCertifications	-	Lista com as *Certifications* atribuídas a cada *User*.
Rule	*DynamicModel*	Uma *Rule* é um objeto que mapeia uma *DEE* rule, que é usada para avaliar ou processar, a algo.
Checklist	*DynamicModel*	Uma *Checklists* é um objeto que contém uma lista de de *Items* que têm que ser processados numa determinada situação.
ChecklistItems	-	Items que fazem parte de cada *Checklist*.
ChecklistItemsParameters	-	Parâmetros de cada Item de uma *Checklist*.
Site	*DynamicModel*	Objeto que permite configurar remotamente o cmNavigo para poder haver troca de informação.
Calendar	*DynamicModel*	Neste contexto, um *Calendar* representa o calendário fiscal de uma empresa e é usado para mapear os dias de um calendário normal com os dias do calendário de operações da empresa.
ShiftDefinition	*DynamicModel*	Este objeto define um turno para uma determinada *Area* e está sempre associado a um *Calendar*.
Facility	*DynamicModel*	Uma *Facility* corresponde à localizaão física de um determinado chão de uma linha de produção.
Area	*DynamicModel*	É uma agregação lógica de *Steps* e *Resources* que está sempre associada a uma *Facility*.
Reason	*DynamicModel*	Define um código ou uma categoria para um evento particular ou uma associação, espeficamente para perdas, bónus, *reworks* e *holds*.
SortRuleSet	*DynamicModel*	É uma coleção de *Sort Rules* que são usadas para distribuir os *Materials* e os *Resources* para fazer *Dispatch*.
Service	*DynamicModel*	Um *Service* é o processo/operação que um *Material* utiliza num determinado *Step*.
Step	*DynamicModel*	Representa um processo/operação de manufatura com determinadas propriedades que um *Material* tem que efetuar.
StepReasons	-	As *Reasons* que um *Step* pode incluir
Flow	*DynamicModel*	Um *Flow* representa uma rota pré-definida (ou caminho) de um *Material*.
FlowStructures	-	As estruturas que fazem parte de um *Flow*
Product	*DynamicModel*	Um *Product* é um objeto que contém as características desejadas para um determinado *Material* depois de este ser processado.

Resource	*DynamicModel*	Representa qualquer entidade que participa no processamento ou armazenamento de *Materials*, tais como equipamentos ou pessoal.
Parameter	*DynamicModel*	É um objeto usado para definir entidades reutilizáveis para *Data Collections*, *Recipes* e *SPC*.
Protocol	*DynamicModel*	Os *Protocols* são usados para tratar exceções.
ProtocolStates	-	Estados que fazem parte dos *Protocols*.
DataCollection	*DynamicModel*	Uma *Data Collection* é um plano para recolher dados quantitativos em determinados pontos do sistema.
DataCollectionParameters	-	Parâmetros de uma *Data Collection*.
DataCollectionLimitSets	*DynamicModel*	Define avisos e limites de erros para uma determinada *Data Collection* num *Step* específico.
BOM	*DynamicModel*	Uma *BOM* (*Bill Of Materials*) representa o produto original com as respetivas quantidades necessárias para montar uma unidade de um determinado *Product* num *Step*.
BOMProducts	-	Tabela com as ligações entre os *Products* e os *BOM*.
Chart	*DynamicModel*	Um Chart é um objeto que representa um SPC Chart.
Recipe	*DynamicModel*	Representa a informação do equipamento necessário para o processamento de um trabalho.
RecipeParameters	-	Parâmetros de uma *Recipe*.
SubRecipes	-	Cada *SubRecipe* que uma *Recipe* pode conter.
SubRecipesParameters	-	Parâmetros de uma *SubRecipe*.
Part	*DynamicModel*	É um objeto que representa items físicos que serão usados durante o procedimento de manutenção.
MaintenancePlan	*DynamicModel*	Representa os planos que são definidos no cmNavigo que entram em ação caso seja necessário executar manutenção de algum *Resource*.
MaintenancePlanActivities	-	Atividades que fazem parte de um *Maintenance Plan*.
MaintenancePlanActivitParts	-	Cada tarefa de uma atividade de um *Maintenance Plan*.
MaintenancePlanInstance	*DynamicModel*	É uma instância do já falado *Maintenance Plan*.
GenericTable	*GenericTable*	*GenericTables* que fazem parte do sistema.
SmartTable	*SmartTable*	*SmartTables* que fazem parte do sistema.

Container	*DynamicModel*	É um objeto que contém pode conter *Materials* e é usado para transportá-los de um ponto para outro da linha de produção.
Material	*DynamicModel*	O *Material* é o centro do sistema. Representa qualquer matéria-prima, inventório ou até um conjunto de *Materials* do mesmo tipo.
Config	*Static Model*	Configurações adicionais do sistema.